HOME · COMPUTER · AKTIV

HCA

Das Magazin für Acht-Bit-Computer

TI REVUE
DAS MAGAZIN FÜR TI 99/4A & TI PC

Book Collection 4

TI REVUE

Das Magazin für TI 99-4A

NR. 2/87

Einfü hrung inGP 'L (II)

Speech-Synthe-Sizer erklärt

Fernschreiber als Drucker

Dialog

Fünf Seiten Börse

DAS HEFT IM HEFT FÜR TI-USER

mispa Microcomputer

bietet an

ASSEMBLER SEMINAR TI 99

vom 01.05.87 bis 03.05.87 veranstalten wir einen Kurs, der Sie in die Grundlagen und Geheimnisse der Maschinenprogrammierung des TI-99/4A einführt.

Fordern Sie nähere Unterlagen kostenlos bei uns an:

mispa Microcomputer, Roland Reinl
Münchner Str.48, 8025 Unterhaching
Telefon 089 / 61 90 48

EINFÜHRUNG IN GPL: NEUE ADRESS-FORMATE

Versprochen ist versprochen: Sie kommen: Nein, nicht Poltergeister, sondern neue Adreßformate. Bestimmt ist Ihnen nämlich auch aufgefallen, daß im letzten Teil ständig das Wort „direkt" so betont war. Das lag daran, daß Sie gleich alles über indirekte Operanden erfahren, zusammen mit einigen neuen Befehlen. Machen Sie es sich in Ihrem Chefsessel bequem, denn die Befehle sind allesamt sehr leicht verdaulich. Schonkost ist heute angesagt, bevor in Folge IV dann einige Bolzen kommen.

ALL 32, der Bildschirm-Ex: ALL füllt alle 768 Positionen des Bildes mit dem angegebenen ASCII-Code, wobei 32 bekanntlich das Leerzeichen ist. Rufen Sie ein GPL-Programm mit dieser Anweisung aus dem Basic heraus durch CALL auf, so müssen Sie wie bei allen Bildschirmoperationen den Screenoffset von +96 beachten, was ALL 128 ergäbe. Die Bytes >837E (YPT) und >837F (XPT) werden dabei gelöscht. Dies ist aber nur für Anweisungen wichtig zu wissen, die auf diese Bytes zurückgreifen, wie z.B. FMT.

Eine unvorhersehbare Sache: RAND

Das Byte >8378 bekommt einen völlig zufälligen Wert zugeornet. Den zulässigen Wertebereich hierfür können Sie frei wählen. Bei RAND 255 z.B. kann es jeden Wert von 0bis 255 annehmen, bei RAND 24 nur jeden von 0 bis 24. Für Neugierige: Wie kommt ein TI zu einer Zufallszahl? Nach dem Einschalten erscheint das Titelbild und die Tastatur wird abgefragt. Auf Tastendruck hin erscheint das Hauptmenü und die Tastatur wird wieder abgefragt. Beide Male ist der RAND-Befehl in die Abfrage eingebaut. Da es nie möglich ist, zweimal mit exakt gleicher Geschwindigkeit durch Titelbild und Hauptmenü zu kommen, wird das RAND-Statement mal mehr, mal weniger oft abgearbeitet und somit steht nach dem Starten eines Programms vom Menü aus immer ein anderer Wert in > 8378.

Mit Exit geht's zurück.

Exit ist ein Software-Reset. Er hat die gleiche Wirkung wie GPLLNK mit DATA > 0020 oder BLWP §> 0000 im Maschinencode, nämlich einen Systemreset auszuführen. Fast jedes GPL-Programm wird am Ende so wieder verlassen.

Arithmetische und logische Befehle:

CLR, ADD, DEC,... und dergleichen mehr

Wir brauchen sie in unserem Beispielprogramm, wir brauchen sie aber praktisch immer, denn kaum ein Programm kommt ohne diese kleinen Alleskönner aus. Addieren, Subtrahieren, Multiplizieren und Dividieren, Aufwärts- und Abwärtszählen oder logische Verknüpfungen herstellen, nichts geht ohne arithmetische und logische Befehle. Sie haben alle den gleichen Aufbau und alle besitzen mindestens einen, aber höchstens zwei Operanden. Diese werden exakt so gehandhabt, wie dies letztesmal bei ST und DST der Fall war. Erinnern Sie sich? Sie lasen, daß ausnahmslos alle (da waren sogar noch drei Rufzeichen dahinter!) arithmetische und logische Befehle, zu denen ST und DST auch dazugehören, sowohl Byte- als auch Wortoperationen durchführen.

Und in der Erklärung zum Beispielprogramm stand dann am Beispiel des DST, daß der zweite Operand bei den Befehlen, die einen solchen haben, sowohl ein IMM-Operand, als auch ein allgemeiner Adreßoperand sein kann. Und daß die Unterscheidung beider mit dem vorletzten Bit im Befehlsbyte geschieht. Und die, die nur einen Operanden haben? Bei ihnen gehört das vorletzte Bit eben noch zum Befehlscode dazu und hat von daher einen festen Wert. Da dies die einfacheren Befehle aus dieser Gruppe sind und sie außerdem noch die Eigenschaft haben, mit den entsprechenden Befehlen des Maschinencodes fast identisch zu sein, kann man es sich leisten, sie einfach mit Verweis auf den Maschinencode tabellarische aufzulisten. Hier sind sie also:

GPL-Befehl	Code (Bin/Hex)	GPL-Statusbyte	Maschinen-Befehl
ABS	1000000. (80–81)	unbeeinflußt	ABS
NEG	1000001. (82–83)	unbeeinflußt	NEG
INV	1000010. (84–85)	unbeeinflußt	INV
CLR	1000011. (86–87)	unbeeinflußt	CLR
INC	1001000. (90–91)	CPU => GPL	INC
DEC	1001001. (92–93)	CPU => GPL	DEC
INCT	1001010. (94–95)	CPU => GPL	INCT
DECT	1001011. (96–97)	CPU => GPL	DECT

Der Ausdruck CPU=>GPL bedeutet, daß das GPL-Statusbyte gleich dem Hi-Byte des CPU-Statusregisters wird, und der Punkt bei den Binärzahlen steht für das Bit, welches die Wort- und Byteverarbeitung unterscheidet. Jeder dieser Befehle steht, wie erwähnt, immer zusammen mit einem allgemeinen Adreßoperanden.

Eine Speicherstelle zeigt auf eine andere: Das Prinzip der indirekten Adressierung

Sie geben eine Adresse an, z.B. VDP§>0001. Der Computer nimmt den Wert aus dieser Adresse, rechnet mit ihm das, was der zugehörige Befehl ihm sagt und legt das Ergebnis zurück in die Adresse. So

wurde es Ihnen in der letzten Folge gezeigt. Was soll sich daran nun ändern? Eigentlich gar nichts, außer, daß zwischendrin noch ein Schritt hinzukommt. Der Computer nimmt den Wert aus der Adresse und rechnet mit ihm nicht. Vielmehr sieht er diesen Wert nun seinerseits als Adresse an, nimmt den Wert aus dieser Adresse, rechnet mit ihm das, was der Befehl ihm sagt und legt das Ergebnis zurück in diese Adresse. Dieses Verfahren kennen Sie übrigens auch vom Maschinencode her: Z.B. *R4. In GPL haben Sie nur viel mehr Möglichkeiten, denn Sie sind nicht an ein Register gebunden, die es in GPL ohnehin nicht gibt, sondern Sie können aus jedem direkten einen indirekten Operanden machen, z.B. VDP*>0001.
Im Klartext sieht das folgendermaßen aus:

V. Der kleine Bereich – Indirekt

Der Operand weist auf ein Byte im Bereich von >8300 bis >91FF des Scratch-Pad-RAMs. Der Inhalt dieses Bytes plus >8300 ergibt die endgültige Adresse im Bereich von >8300 bis >83FF. Das Format entspricht dem Format II, das 4. Bit ist aber gesetzt.

Beispiel: Den Operanden *>8317 beschreiben Sie mit >9017.
Format: 1001 ++++ ....
* Adresse *

VI. Der gesamte Bereich – Indirekt

Der Operand weist auf jedes beliebige Byte, dessen Wert >8300 wieder die endgültige Adresse im Bereich von >8300 bis >83FF ergibt. Das Format entspricht dem Format III mit dem gesetzten 4. Bit, auch hier darf der >8300-Offset nicht vernachlässigt werden.

Beispiel: Den Operanden *>1742 beschreiben Sie mit >9F>9442
Format: 1001 1111 ....
++++ *Adresse
–> 8300*

VII. Der VDP-Bereich – Indirekt

Sie benutzen das Format V oder VI, setzen das 3. Bit auf 1 und bekommen so den Operanden für indirekte VDP-Adressierung. Er zeigt diesmal auf ein Wort (!), dessen Wert wiederum die eigentliche VDP-Adresse (von >0000 bis >3FFF ohne Offset) ist.

Beispiel 1: Sie geben für VDP*>8317 die Hexcodes >B017 ein.
Format VII/1: 1011 ++++ ....
* Adresse *

Beispiel 2: Sie geben für VDP*>1742 die Hexcodes >BF>9442 ein.
Format VII/2: 1011 1111 ....
++++ * Adresse
–> 8300 *

Das war's praktisch schon für heute. Aber nichts für ungut, nächstes Mal geht's weiter mit indizierten Operanden. Zum Schluß noch .. Na, raten Sie mal!

... das Programm

Eine Zufallsgrafik soll es diesmal werden. 30 Sterne werden wahllos auf dem Bildschirm verstreut, nach kurzer Zeit wieder gelöscht und aufs Neue verteilt.
Alles mit ein wenig Sounduntermalung, um den I/O nochmal etwas aufzufrischen. Schließlich sind wir ja jetzt schon in der Lage, etwas ausführlichere Programme zu schreiben, wenn auch noch nichts Umwerfendes. Beginnen wir mit dem Üblichen:

```
G6000 AA00   DATA >AA00
G6002 0000   DATA >0000
G6004 0000   DATA >0000
G6006 6010   DATA >6010
G6008 0000   DATA >0000
G600A0000    DATA >0000
G600C 0000   DATA >0000
G600E 0000   DATA >0000
G6010 0000   DATA >0000
G6012 601B   DATA >601B
G6014 06     BYTE >06
G6015 5354   TEXT 'STERNE'
      4552
      4E45
```

Setzen Sie zuerst die Hintergrundfarbe, z.B. nach Weiß:

```
G601B 040F   BACK 15
```

Damit benötigen wir einen sauberen Bildschirm. Er wird zwar schon beim Aufrufen unseres Programms vom Menü aus automatisch gelöscht, da diese Routine aber in einer Schleife ständig wiederholt werden soll, müßte er jedesmal aufs Neue gelöscht werden.
ALL hat den Code >07 und wie BACK den IMM-Wert als 2. Byte:

```
G601D 0720   ALL 32
```

Danach setzen Sie den Wert für die Schleife. 30mal soll der Rechner sie durchlaufen. Und wie Sie vielleicht noch wissen, haben Sie das Scratch-Pad-RAM von >8300 bis >8317 frei zur vorübergehenden Verfügung. Weisen Sie >8300 die Anzahl der gewünschten Durchläufe zu, also den Wert 30 bzw. >1E (hex):

```
G601F BE00   ST §>8300,30
      1E
```

Jetzt ist Ihr scharfer Verstand gefragt. Sie benötigen einen zufälligen Wert zwischen 0 und 767, denn so viele Positionen besitzt der Bildschirm (24 Zeilen mal 32 Spalten), die RAND-Anweisung gibt uns jedoch nur maximal Werte zwischen 0 und 255.
Was tun? Sehen wir uns die Situation in Hex-Ziffern an. Sie benötigen eine Zahl zwischen >0000 und >02FF, bekommen aber nur solche von >00 bis >FF.
Was liegt also näher, als >02FF in ein Hi-Byte und ein Low-Byte zu teilen, die getrennt ihren Wert bekommen? Das Hi-Byte einen von >00 bis >02, das Low-Byte einen von >00 bis >FF. RAND besitzt den Befehlscode >02 und ebenfalls den IMM-Wert als 2. Byte:

```
G6022 0202   RAND >02
G6024 BC01   ST §>8301,
             §>8373
      78
G6027 02FF   RAND>FF
G6029 BC02   ST §>8302,
             §>8373
      78
```

Das Wort bei >8301 enthält nun die Zufallszahl von 0 bis 767, wie Sie es wollten. Damit den Bildschirm zu adressieren, das funktioniert nur mit indirekter Adressierung. Der Operand weist auf >8301 und dort findet der GPL-Interpreter die echte Adresse:

```
G602C BEB0   ST VDP*>8301,42
      012A
```

42 ist dabei der ASCII-Code des Sterns (>2A hex).

Wie gesagt, etwas Sound soll auch dabei sein. Lange Erläuterungen sind hier inzwischen wohl überflüssig, was Soundverarbeitung betrifft.
Die Speicherstellen >8301/2 haben ihre Schuldigkeit getan, somit können Sie sie für den I/O überschreiben:

```
G6030 BF02   DST §>8302,
             >6056
      6056
G6034 F602   I/O §>8302,>00
      00
```

Die folgende Warteschleife soll den TI etwas beschäftigen, solange dieser den Ton erzeugt:

```
G6037 8602   CLR §>8302
G6039 9202   DEC §>8302
G603B 4039   BR GROM§>6039
```

Da Sie bestimmt das Programm auch irgendwann wieder verlassen wollen (ohne FCTN +), ist eine entsprechende Tastaturabfrage notwendig. EXIT besteht aus nur einem Byte: >0B

```
G603D 03     SCAN
G603E 4041   BR GROM§>6041
G6040 0B     EXIT
```

Jetzt müßten Sie nur noch die Schleife bilden und wären praktisch fertig. Sie zählen >8300 so lange herunter, bis Null erreicht ist. Dabei kommt Ihnen eine wichtige Eigenschaft des DEC zu Hilfe. Wie Sie der Tabelle im Abschnitt über arithmetische und logische Befehle entnehmen können, kopiert der DEC den CPU-Status ins GPL-Statusbyte. Das EQ-Bit des CPU-Status kommt dabei nämlich genau auf das C-Bit des GPL-Status zu liegen, was wirklich sehr günstig ist, da Sie an dieses mit BR und BS heran können:

```
G6041 9200  DEC §>8300
G6043 4022  BR GROM§>6022
```

Ist nach dem DEC der Wert in >8300 nicht Null, wird das EQ-Bit nicht gesetzt, damit ist natürlich auch das C-Bit nicht gesetzt und BR führt zur Positionsbestimmung des nächsten Sternchens zurück. Diese Eigenschaft des DEC haben wir uns übrigens in der Warteschleife bei G6037 schon stillschweigend zunutze gemacht.
Lassen wir noch einen Schlußton ertönen und nach einer kurzen Pause die Routine neu beginnen:

```
G6045 BF02  DST §>8302,
            >605B
      605B
G6049 F602  I/O §>8302,>00
      00
G604C BF02  DST §>8302,
            >0A00
      0A00
G6050 9302  DDEC §>8302
G6052 4050  BR GROM§>6050
G6054 401D  BR GROM§>601D
```

Mit der Soundliste wäre Ihr Programm dann vollständig:

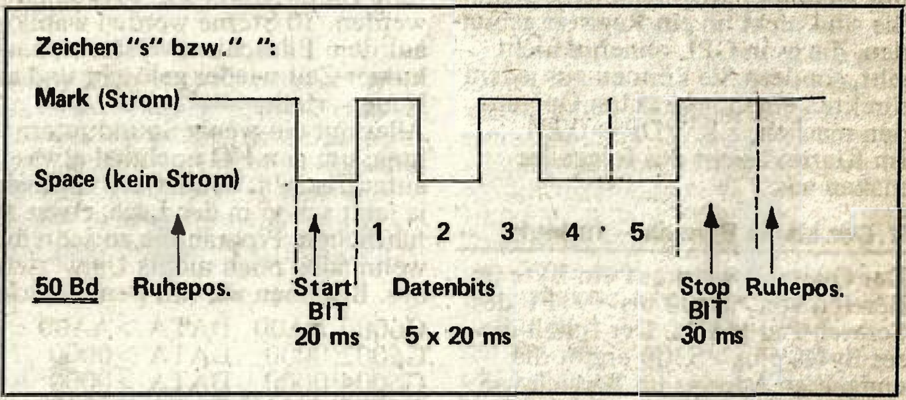
```
G6056 03 9F E2 F5 09
G605B 04 8C 33 90 FF 3C
```

Halt! Fehlen da nicht die Schlußbytes der Liste? Ja, aber sie sind ausnahmsweise diesmal in der Tat unnötig. Solange Sie nämlich sicherstellen, daß, noch während der eine Ton läuft, der nächste bereits gestartet wird, die Interruptsoundroutine das Ende der Liste also garantiert niemals erreicht, wird diese auch nie Gelegenheit haben, die Schlußbytes zu verlangen. Konkret heißt das, daß Sie die beiden Schleifen nach den I/O-Statements nicht zu groß dimensionieren dürfen, da Sie sonst wirklich erheblichen Ärger bekämen. Sie dürfen aber auch nicht zu klein sein, sonst haben Sie nichts von dem Ton. Noch ehe er richtig erklingen würde, hätte der TI schon den nächsten I/O erreicht.
Bis dann, auf ein errorloses Wiedersehen!

Sven Dyroff

TTYOUT

Die in Europa verbreiteten Fernschreiber arbeiten, soweit sie für den Telex-Verkehr vorgesehen sind, mit 50 Baud und dem internationalen Telegraphenalphabet No.2, das den 5-Bit Baudot-Code beinhaltet.
Bei dieser Datenübertragungsrate ergibt sich für ein zu übertragendes Zeichen folgende Impulsfolge:



Um eine andere Baudrate einzustellen muß man folgende DATAs in Programmzeile 330 ändern: (unterstrichen)

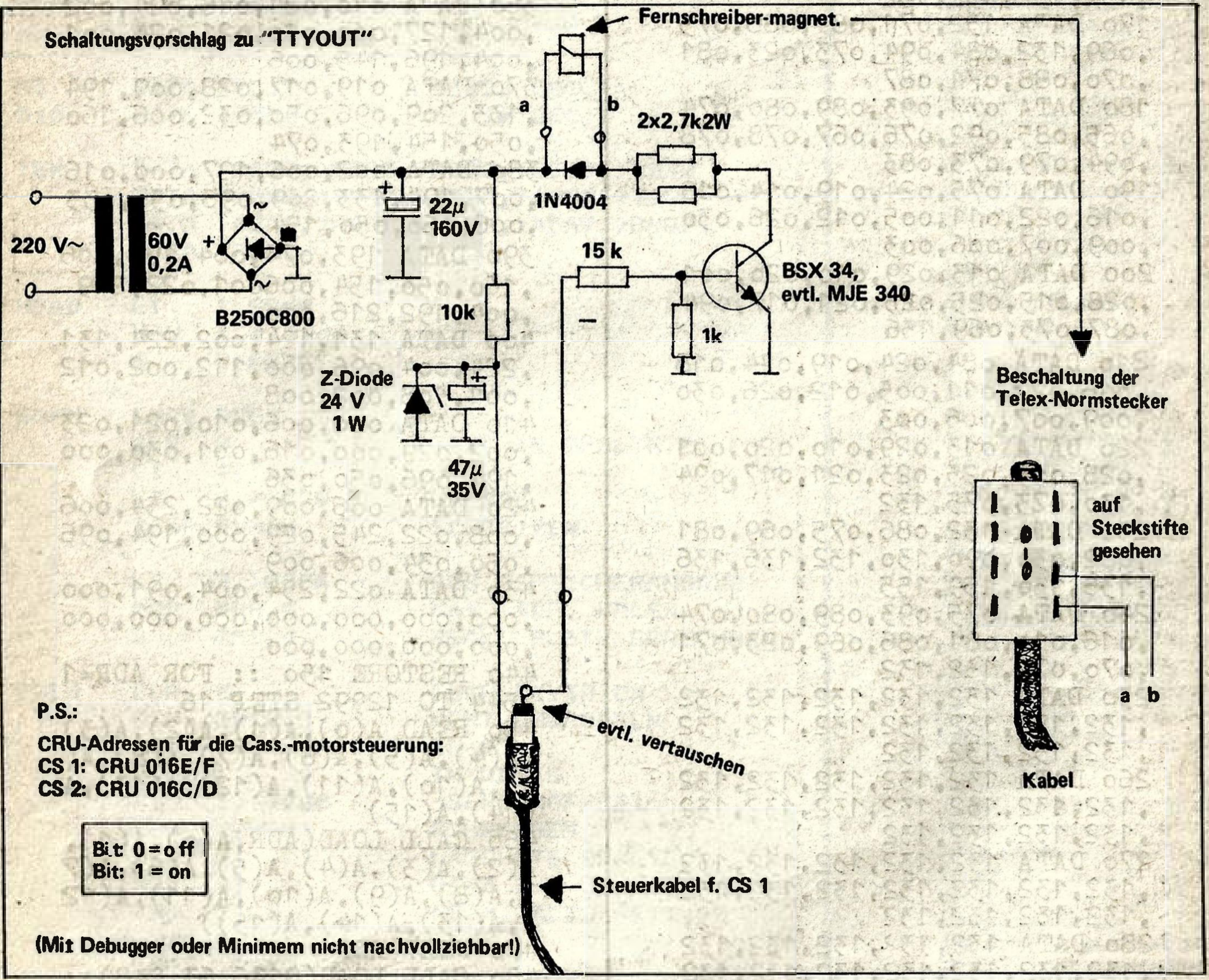
330 DATA 108,124,<u>009,000,006,000</u>,...

stopbit dauer — datenbit dauer

Der richtige Wert muß durch Versuche ermittelt werden, für 45,45 Bd. (RTTY, US-Fernschreiber) sind z.B. etwas größere Werte (längere Bitdauer= langsamere Übertragung) einzusetzen. Hierbei ist zu beachten, daß es auch Fernschreiber gibt, deren Stopbitlänge gleich der Datenbitlänge ist.
Der vom Programm erzeugte Code wird als Folge von „geöffnet" und „geschlossen" des Rekordersteuerungstransistorschalters ausgegeben. Um den Fernschreiber damit zu betreiben, ist ein Linienstromnetzteil (50 mA/ 100 V=), ein Reihenwiderstand (ca. 2 kΩ 10 Watt) und ein ausreichender Schalttransistor notwendig, zur Potentialtrennung Computer/Fernschreiber dient ein Optokoppler.
Da auf einer Fernschreibmaschine nur ein Bruchteil des ASCII-Zeichensatzes verfügbar ist (nur Kleinbuchstaben, wenige Zeichen) ist die Codewandlung insofern unvollkommen, als verschiedenen ASCII-Charactern ein- und dasselbe Fernschreibzeichen zugeordnet wird, unter Umständen ohne sinnvollen Zusammenhang. Um das Programm absturzsicher zu machen, sind im zu übergebenden String alle ASCII-Codes zwischen 0 und 255 erlaubt, bzw. ergeben ein gültiges Fernschreibzeichen. (Gegebenenfalls halt nur "Space".)
Die zur Druckformatierung wie auch zum einfachen „rausnudeln" notwendigen Steuerzeichen wie Zeilenvorschub und Wagenrücklauf (Carriage Return & Line Feed) sind sowohl auf die entsprechenden

ASCII-Codes als auch auf mehrere druckbare Sonderzeichen des TI-99 gelegt. So ergibt zum Beispiel bei ASCII 95 (Unterstrich, FCTN U) ein LINE FEED und ASCII 125 (Senkrechte Striche, FCTN A) ein CARRIAGE RETURN, diese kleine Besonderheit erspart einem beim Erstellen auszudruckender Texte das dauernde “&CHR$(..)&“ ...
Dem eigentlichen Programm geht in der Pokeliste die wesentlich längere Codetafel voran, in der jedem ASCII-Code ein interner 8-Bit-Code zugeordnet wird, der neben dem Startbit und den 5 Datenbits jedes Baudotcharacters auch die Information über die Zugehörigkeit zu den „Zahlen“ oder „Buchstaben“ – oder zu beiden (z.B. CARRIAGE RETURN oder SPACE) enthält, die zur automatischen Generierung der Umschaltzeichen unerläßlich ist.
Daß viele ASCII-Zeichen vom Fernschreiber nicht gedruckt werden können, wird durch den konkurrenz-

```
100 rem ++++++++++++++++++++++
101 rem + TTYOUT-POKELISTE   +
102 rem +                    +
103 rem + VERSION 3.1        +
104 rem +                    +
105 rem + ENVIRONMENT:       +
106 rem + TI-99/4A           +
107 rem + EXTENDED BASIC     +
108 rem + 32k-EXPANSION      +
109 rem +                    +
110 rem + DESCRIPTION:       +
111 rem +      CALL LINK     +
112 rem + ("TTYOUT",M$)      +
113 rem + DELIVERS 50-BAUD   +
114 rem + TELETYPE-CODE AT   +
```



los günstigen Preis eines solchen „Druckers“ relativiert.
Die Ansteuerschaltung sollte im Übrigen von jemandem aufgebaut werden, der sich etwas aufs Elektronische versteht – das Linienstromnetzteil ist nicht ganz ohne und beim notwendigen Herumprobieren sollte sich der Linienstrom nicht in die Computerkonsole verirren...

```
115 rem + CASSETTE-CONTROL   +
116 rem + (BLACK WIRE) OF    +
117 rem + CS1.               +
118 rem + CCIR 2 ALPHABET    +
119 rem + IS USED, ASCII     +
120 rem + CHARACTERS ARE     +
121 rem + AUTOMATICALY       +
```

```
122 rem + TRANSCRIPTED,AS  +
123 rem + THIS IS POSSIBLE.+
124 rem ++++++++++++++++++++
125 rem + PROGRAM WRITTEN  +
126 rem + BY MARTIN SOMMER +
127                        +
128 rem ++++++++++++++++++++
130 CALL INIT
140 DIM A(15)
150 DATA 132,132,132,132,132
,132,132,132,130,132,136,136
,132,130,155,159
160 DATA 132,132,132,132,132
,132,132,132,132,132,132,132
,132,132,132,132
170 DATA 132,071,084,086,075
,069,132,084,094,073,023,081
,070,088,071,087
180 DATA 077,093;089,080,074
,065,085,092,076,067,078,078
,094,079,073,083
190 DATA 075,024,019,014,018
,016,022,011,005,012,026,030
,009,007,006,003
200 DATA 013,029,010,020,001
,028,015,025,023,021,017,094
,087,073,069,136
210 DATA 084,024,019,014,018
,016,022,011,005,012,026,030
,009,007,006,003
220 DATA 013,029,010,020,001
,028,015,025,023,021,017,094
,130,073,075,132
230 DATA 132,086,075,069,081
,082,030,090,130,132,136,136
,136,130,159,155
240 DATA 075,093,089,080,074
,016,010,081,086,069,023,071
,070,079,132,132
250 DATA 132,132,132,132,132
,132,132,132,132,132,132,132
,132,132,132,132
260 DATA 132,132,132,132,132
,132,132,132,132,132,132,132
,132,132,132,132
270 DATA 132,132,132,132,132
,132,132,132,132,132,132,132
,132,132,132,132
280 DATA 132,132,132,132,132
,132,132,132,132,132,132,132
,132,132,132,132
290 DATA 132,132,132,132,132
,132,132,132,132,132,132,132
,132,132,132,132
300 DATA 132,132,132,132,132
,132,132,132,132,132,132,132
,132,132,132,132
310 DATA 000,000,000,000,000
,000,000,000,000,000,000,000
,000,000,000,000
320 DATA 000,000,000,000,000
,000,000,000,000,000,000,000
,000,000,000,000
330 DATA 108,124,009,000,006
,000,002,224,050,000,004,192
,002,001,000,001
340 DATA 002,002,048,000,007
,018,004,032,032,020,004,193
,192,194,208,115
350 DATA 006,193,004,197,209
,115,006,197,209,101,049,000
,010,021,023,002
360 DATA 010,021,016,024,002
,004,127,000,010,021,024,001
,004,196,145,006
370 DATA 019,017,028,009,194
,133,209,096,050,032,006,160
,050,154,193,074
380 DATA 002,006,127,000,016
,007,194,133,209,096,050,033
,006,160,050,154
390 DATA 193,074,004,198,006
,160,050,154,006001,022,219
,004,192,216,000
400 DATA 131,124,002,224,131
,224,004,096,000,112,002,012
,001,108,002,008
410 DATA 000,006,010,021,023
,002,029,000,016,001,030,000
,194,096,050,036
420 DATA 006,009,022,254,006
,008,022,245,029,000,194,096
,050,034,006,009
430 DATA 022,254,004,091,000
,000,000,000,000,000,000,000
,000,000,000,000
440 RESTORE 150 :: FOR ADR=1
2544 TO 12992 STEP 16
450 READ A(0),A(1),A(2),A(3)
,A(4),A(5),A(6),A(7),A(8),A(
9),A(10),A(11),A(12),A(13),A
(14),A(15)
460 CALL LOAD(ADR,A(0),A(1),
A(2),A(3),A(4),A(5),A(6),A(7
),A(8),A(9),A(10),A(11),A(12
),A(13),A(14),A(15))
470 NEXT ADR
480 CALL LOAD(8196,63,248)::
 CALL LOAD(8274,48,0)
490 M§="TYOUT"&CHR§(50)&CHR
§(38)
500 FOR I=1 TO 8 :: CALL LOA
D(16375+I,ASC(SEG§(M§,I,1)))
:: NEXT I
510 END
```

```
* *********************************
* *           HEADLINE            *
* *                               *
* *         VON W.SOMMER          *
* *                               *
* * START: CALL LINK("HEAD")      *
* * STOP : CALL LINK("STOP")      *
* *********************************

        DEF   HEAD,STOP

        AORG  >24F4

REGS    BSS   >20
VMBW    EQU   >2024
VSBW    EQU   >2020
HH      EQU   >8387

MI      DATA  >A03F           41023 DEC.
D10     DATA  >000A              10 DEC.

TEX     DATA  >8080,>8080,>8080,>8080,>8080
        DATA  >8080,>80A2,>B9B4,>A587,>B380
        DATA  >B2A1,>AD80,>A6B2,>A5A9,>8080
        DATA  >8080

HEAD    LI    1,SOMM
        MOV   1,@>83C4
        B     @>0070

SOMM    LWPI  REGS
        CLR   0               TEXT IN SPALTE 1
        LI    1,TEX
        LI    2,>0020         LAENGE
        BLWP  @VMBW           SCHREIBEN

        LI    0,>000A         POS. SPEICHERANGABE
        MOV   @HH,4           MAX. ADR. HOLEN
        S     @MI,4           REAL. PLATZ ERRECHNEN

BILD    CLR   3               FUER DIVISION
        DIV   @D10,3          WERT UMWANDELN
        MOV   4,1             IN R1
        SWPB  1               IN HBYTE
        AI    1,>9000         -SCREENOFFSET
        BLWP  @VSBW           SCHREIBEN
        DEC   0               NAECHSTE SPALTE
        MOV   3,4             R4=0 ?
        JNE   BILD            NEIN, ALSO REST<>0

        LWPI  >83E0
        RT

STOP    CLR   @>83C4
        B     @>0070

        END
```

TURBOTAPE FÜR DEN TI

1.) Neulich stieß ich beim Durchblättern einer Computerzeitschrift aus dem Jahre 1984, in einem Vergleich der damals bekanntesten Computer, beim TI in der Rubrik Externe Speicher auf den Begriff „Wafertape". Was ist dies und wie arbeitet es?
2.) In einem Prospekt von TI sah ich einen Drucker derselben Firma mit dem Namen TI-Impactdrucker für den TI-99/4A. Ist dieser noch erhältlich? Kann man ihn direkt an den TI anschließen, d.h., ohne Interface?
3.) Ist es möglich, ein Programm in TI-BASIC direkt nach dem Laden, also ohne extra RUN einzugeben, zu starten?
4.) Gibt es ein Turbotape für den TI in TI-BASIC?

Christian Oßwald, Kirchheim

Ein Wafertape ist in Funktion und Aufbau mit einem Kassettenrekorder zu vergleichen. Der einzige Unterschied besteht darin, daß dieses mit Endloskassetten betrieben wird.
Beim TI-Impaktdrucker von Texas Instruments handelt es sich um einen Epson MX80 mit aufgedruckten TI-Namen. Dieser Drucker ist nur via RS232 bzw. Centronics zu betreiben. Er ist nicht mehr im Handel als „TI-Impactdrucker" erhältlich.
Es ist, ohne entsprechende Erweiterung, nicht möglich, ein TI-BASIC-Programm, ähnlich wie RUN"CS1" in X-BASIC, direkt zu starten.
Ein Schaltungsvorschlag eines Turbotapes wurde in einer älteren Ausgabe des Computermagazins CHIP veröffentlicht.

RAM-DISK AM TI

Ich besitze einen TI-99/4A mit X-BASIC und

32k-Erweiterung.
Nun las ich in TI-REVUE 11/86 „... ein anderes Peripheriegerät zu verwenden, das MERGE erlaubt, z.B. eine RAM-Disk." Haben Sie ein derartiges Programm, das MERGE ohne Diskettenlaufwerk, also mit Kassette, erlaubt?
Horst Caspers, Mosbach

Bei einer RAM-Disk handelt es sich um ein Programm, das einen bestimmten Speicherbereich so nutzt, als sei es ein Diskettenlaufwerk. Da wir kein Programm besitzen, das MERGE von Kassette erlaubt, geben wir Ihre Frage an die Leser der TI-REVUE weiter. Sollte jemand ein solches Programm besitzen, so kann er es uns zusenden, wir leiten es dann an Sie weiter.

AMERIKANISCHE GERÄTE FÜR DEUTSCHLAND UMBAUEN

Sie hatten in einer älteren TI-REVUE beschrieben, daß der USA TI-99/4A für das deutsche Heim umgebaut werden muß. Bei welchen Produkten muß ich auch Änderungen vornehmen, daß diese auch in Deutschland richtig funktionieren?
Andre Schmiedl, Stemwede

Sie müssen an allen Geräten, die ihren Strom vom Netz her, also via Computer oder anderer Stromquelle, beziehen, eine Änderung vornehmen. Hierbei muß nämlich der Transformator von den amerikanischen 110 V auf die deutsche Norm, also auf 220 V, 50 Hz, umgestellt werden. Die davon betroffenen Geräte sind die Konsole, der Drucker, die P-Box und die Modems. Bei den Modems müssen Sie zusätzlich noch darauf achten, daß sie der deutschen Bundespost entsprechend mit einer FTZ-Nummer versehen sind. Bei der Konsole muß zusätzlich noch der PAL-Modulator ausgetauscht werden, da in Deutschland 50 Hz bei den Fernsehgeräten Norm sind.

TI-WRITER FILES ENTSCHLÜSSELN

Ich möchte gerne eine mittels TI-Writer erstellte Datei z.B. alphabetisch ordnen und lese dazu diese Datei in ein TI-BASIC-Programm ein. Wenn die eingelesene Datei ausgedruckt, am Bildschirm oder mit dem TI-Writer dargestellt wird, erscheint sie aber unvollständig.
Erzeugt man eine „schwebende Datei" durch ein Komma hinter dem INPUT-Befehl, so fehlen Kommata im Text und der String wird ab dem (fehlenden) Komma auf der nächsten Zeile fortgesetzt.
Läßt man das Komma hinter dem INPUT-Befehl weg, so fehlt der Rest ab dem nicht dargestellten Komma. Was ist zu tun, damit eine fehlerfreie Bearbeitung der Sätze auch mit Kommata erfolgt?
Dann ergibt sich noch eine weitere Schwierigkeit im Druckformat (Tabulator, rechter- bzw. linker Rand etc.). Diese Angaben werden im letzten Datensatz festgehalten. Wie sind diese verschlüsselt, und wie kann man sie ändern?
Eberhard Schuon, Eningen

Ein korrektes Einlesen der Datensätze, sofern sie Kommata enthalten, ist nur in X-BASIC mittels LINPUT möglich. Dieser Befehl akzeptiert ein Komma innerhalb eines Strings. Im Gegensatz dazu sieht der Befehl INPUT ein Komma als Trennung zwischen zwei Strings an. Somit wird ab dem Komma der String entweder abgeschnitten, oder, im Falle einer schwebenden Datei, der Rest als neuer String eingelesen.
Die Art, mit der der TI-Writer seine Steuerzeichen verschlüsselt, ist nicht bekannt. Wir raten Ihnen daher, es einfach auszuprobieren. Vielleicht kann Ihnen auch ein TI-REVUE-Leser in diesem Fall weiterhelfen.

RECHTUND ORDNUNG

Sind bei einem Programm, das in einer Computerzeitschrift veröffentlicht worden ist, auch Musik, Charaktere etc. urheberrechtlich geschützt?
Kann man ein Programm, das man in einer Computerzeitschrift veröffentlicht hat, noch zusätzlich unter der Freeware-Option vertreiben?
Ralph Benzinger, Mannheim

Ihre urheberrechtlichen Fragen sind etwas schwierig zu beantworten, da es hier stark auf den Einzelfall ankommt.
Bezüglich des Vertriebs von eigenen, veröffentlichten Programmen unter der Freeware Option kommt es stark auf die Geschäftsbedingungen der jeweiligen Zeitschrift an. Im Zweifelsfall empfiehlt es sich, schon bei der Einsendung des Programmes an eine Zeitschrift um Klärung dieser Frage zu bitten. Bei dem urheberrechtlichen Schutzes von Teilen eines Programmes kommt es ebenfalls auf den Einzelfall an. Sie können grundsätzlich davon ausgehen, daß z.B. einzelne Charaktere wohl kaum zu schützen sind. Bei Musik wird es etwas schwieriger. Bei Programmen, die rein zur Erzeugung von Musik dienen, ist diese wohl geschützt. Kurze Untermalungen sind wohl nicht schützbar. Allerdings ist hier auch noch der reine Schutz einer Komposition zu beachten.

EINGABEHILFE VON CTRL.-CHARAKTERN

Ich habe festgestellt, daß sich das in Ihrer Zeitschrift dargestellte Spiel „Das Labyrinth of Ox" auf folgende Art manipulieren läßt:
Nachdem man die drei Gegenstände im oberen Teil der Höhle eingesammelt hat, durchbreche man eine der beiden Seitenwände und lasse den Bildschirm so lange nach oben scrollen, bis man den Höhlenausgang sieht. Jedes der Zeichen außerhalb des Randes wird vom Computer als eingesammelter Gegenstand gewertet. Auf der Höhe des Ausgangs kann man wieder die Seitenwand durchbrechen und die Höhle betreten.
Außerdem habe ich noch eine Eingabehilfe für die Zeichen CTRL A bis CTRL O in X-BASIC entwickelt. Nach Eingabe der Zeile
FOR A=97 TO 111 : : CALL CHARPAT(A, AS) : : CALL CHAR (A+32,SEGS(AS,5,12) "007C"):: NXT A
sind sie als unterstrichene Buchstaben sichtbar.
Jürgen Toelke, Emmendingen

Vielen Dank für Ihre Tips, vielleicht helfen sie einigen Lesern der TI-REVUE weiter.

ANDERE TI-TASTATUR

In TI-REVUE 12/86 fand ich den Bericht über die Alpha Lock-Taste. Als ich Ihrer Anleitung entsprechend Pin 6 unterbrach und die Diode einsetzte, fielen die Tasten Q, W, E, R und Taus. Nach einigen Überprüfungen stellte ich fest, daß an meinem (von 1981) Pin 10 statt Pin 6 unter-

brochen werden mußte, und die Diode dort einzulöten war. Ist ein Fehler in Ihrer Anleitung, oder habe ich eine andere Tastaturbelegung?
Bernd Stenner, Haltern

Es ist uns nicht bekannt, daß andere Versionen der Tastatur existieren. Der Ausfall der Tasten kann nur von einem Fehler Ihrerseits herstammen. Es besteht die Möglichkeit, daß Sie Pin 15 mit Pin 1 verwechselten und somit die Diode zuerst an Pin 10 anlöteten, und dann erst an Pin 6.

CLUB-ECKE

Diesmal ist die Clubecke sehr kurz. Nur der TI-Club Berlin-West stellt sich vor.

Im November 1986 hat sich in Berlin eine neue TI-Gruppe gebildet, die zwar relativ klein ist, aber sich vorgenommen hat, effektiver zu arbeiten, als dies in einer sehr großen Gruppe möglich ist. Unsere Mitgliederzahl beträgt z.Zt. 8 und wir sind nicht bestrebt, die Gruppe wesentlich zu vergrößern.
Wir befassen uns derzeit mit allen möglichen Dingen, so wird hier in Basic, Maschine und GPL programmiert, auch erste Schritte in <C> haben wir gemacht. Auf der Hardwareseite sind einige neue Dinge in Erprobung.
Unser Bestreben ist es, einen möglichst engen Kontakt zu anderen Clubs und einzelnen TI-Usern zu halten. Das gilt auch für unsere Berliner Freunde.

Wir haben uns entschlossen, als Einstieg eine Diskette mit Instances für den TI-Artist als Freeware an alle Interessierten abzugeben. Leerdiskette und frankierter Rückumschlag genügt, in ein paar Tagen wird die Diskette bei euch ankommen.
Und wer Interesse hat, den C-Compiler von C. Pulley auch für die Gramsimulation zu bekommen, der verfahre wie oben, doch dann bitte zwei Leerdisks.
Nun hoffen wir auf Reaktionen von euch, aber wir wollen natürlich noch unsere Kontaktanschrift mit auf den Weg geben:
TI-Gruppe Berlin-West
Wederstraße 59
1000 Berlin 47

UNSER TELEFONSERVICE

Wie immer steht unseren Lesern unser Telefon-Service zur Verfügung! Jeden Dienstag von 15 bis 19 Uhr. Listings/Programme: 0 89 / 129 80 13

TITEL AUS DEM VERLAG

DER SPRACHSYNTHESIZER DES TI-99/4A ENDLICH ERKLÄRT!

Der Sprachsynthesizer des TI-99/4A ist schon seit langer Zeit bekannt für seine deutliche, saubere, naturgetreue Aussprache. Allerdings ist der Benutzer auf ein zwar nicht gerade kleines Vokabular angewiesen, bei dem man aber doch schnell an seine Grenzen stößt. Ein anderes Problem ist, daß die Wörter einzeln ganz gut klingen, zusammen aber keinen flüssigen Satz ergeben. Dies hat zwei Gründe: Zum einen sind die Wörter, da sie im ROM des Synthesizers untergebracht sind, unveränderbar und klingen jedesmal in derselben Weise. Zum anderen klingen die Wörter alle einzeln aus, aber jeder gesprochene Satz darf erst am Ende ausklingen. Eigentlich schade, denn der Sprachsynthesizer scheint doch, wenn man sich ein paar Kostproben des Vokabulars anhört, ganz brauchbar zu sein.

Von TI geplante Erweiterungen in Form von Steckmodulen, die Platz unter der Klappe des Gerätes hätten finden sollen, sind – aus welchen Gründen auch immer – nie auf dem Markt erschienen. Ein wirklicher Fortschritt im Hinblick auf freie Sprachprogrammierung ist der Terminal Emulator II. Dieser beinhaltet neben der Telekommunikation auch einen Textkonverter, mit dessen Hilfe ganze Sätze aussprechbar werden und beliebig viele Worte verfügbar sind. Wer dieses Modul kennt, wird die allerdings gesunkene Sprachqualität kennen. Die Sätze werden wirklich flüssig ausgesprochen, aber geradewegs ohne hoch und runter. Die Sprache wird steif und maschinell. Damit dem wenigstens etwas abgeholfen werden kann, gab TI die Möglichkeit, Betonungen zu setzen und andere Tonhöhen zu wählen. Aber, ehrlich gesagt, viel Fortschritt war dies im Grunde doch nicht, denn man kann nicht voraussetzen, daß alle, die einen Sprachsynthesizer haben, auch den Terminal Emulator II besitzen. Manche Spiele von TI beinhalten stellenweise exzellente Sprachausgaben, und man kann neidisch werden, daß einem überhaupt nicht möglich ist, solch eine Sprachprogrammierung zu betreiben. Die einzige Möglichkeit, Aufschluß über die Zusammensetzung der Sprache zu erhalten, ist die Ausführung des CALL SPGET-Befehls in Extended Basic. Was man erhält, ist ein ganz verrückter String, von dem man allerhöchstens folgende Information erhalten kann: 1. Zeichen: CHR$(96); 2. Zeichen: CHR$(0); 3. Zeichen: Länge des folgenden Strings. Und hier hört es schon mit einem Schlag auf. Jeder, der versucht, Laute zu isolieren, erlebt herbe Enttäuschungen mit völlig „verkorkstem" Geblubber und Gequietsche! Keine Chance ... Warum?

Auf Seite 351 der Editor/Assembler-Anleitung weist Texas Instruments auf das 'TMS5200 Voice Synthesis Processor Data Manual' hin. Kurzum, ich schrieb TI Inc. in Dallas an und erhielt tatsächlich das genannte Schriftstück, aus welchem ich sämtliche nun folgende Informationen bezog. Mittels dieser Anleitung ist mir die Erstellung der zwei Programme zum Entziffern und Codieren von Sprache gelungen.

Die wesentlichen Teile des Synthesizers sind zum einen der Prozessor, Voice Synthesis Processor (VSP) genannt, zum anderen der ROM-Speicher, Voice Synthesis Memory (VSM). Die Sprache, welche entweder direkt vom Benutzer oder vom VSM (=Vokabular) kommt, ist kodiert, um eine rationelle Speichernutzung zu gewährleisten. Aufgabe des VSP ist es, diesen Code zu entschlüsseln und entsprechende Anweisungen an den D/A-Wandler (welcher im Endeffekt den Ton erzeugt) zu geben.

KEIN GEHEIMNISVOLLER CODE

Der Code ist nichts Geheimnisvolles, er ist lediglich ein Filter, mit dem eine Art Hüllkurve erstellt wird. Grundsätzliche Bestandteile der Sprache sind stimmvolle Laute, wie alle Vokale und auch viele Konsonanten (!), und stimmlose Laute wie die übrigen Konsonanten. (Sprechen Sie einmal das 'M' und dann das 'P' ohne den 'e'-Laut jeweils aus – Sie hören den Unterschied beider Konsonanten.) Durch den Code sind geringe Datenmengen möglich; um die Kommunikation zwischen Konsolenprozessor (CPU) und VSP zu vereinfachen, kann entweder das CPU den VSP durch Statusabfrage testen oder auf Interrupts vom VSP warten. Es ist somit möglich, bis zu 40 Filter in einer Sekunde zu verarbeiten. Das 'Interface' zwischen beiden Prozessoren (CPU und VSP) wird gebildet durch die acht Leitungen des Datenbusses (D0–D7), einer Leitung zum Ankündigen eines Lesevorgangs des CPUs (Read Select, RS) einer ähnlichen beim Schreiben (Write Select, WS), einer Synchronisation (READY) und einer Interruptleitung (INT). Die Synchronisation ist notwendig, weil der VSM nicht so schnell die Daten zum VSP bringt, daß dieser mit dem CPU und dessen Anweisungen mithalten kann. Interrupts werden erzeugt bei Ende der Sprachausgabe, wenn der Puffer nur noch halb voll ist und wenn dieser ganz ausläuft. Doch zum Puffer später.

ADRESSE IMMER 20 BITS GROSS

Das Vokabular, der VSM, ist ein TMS6100-Baustein. Bis zu 16 Stück desselben können parallel gelegt werden, um den Speicher hiermit auf 256K-Byte zu erweitern. Die Adressierung des VSM geschieht über vier Adreßleitungen. Die Adresse ist immer 20 Bits groß: 4 Bits für Chip-Selektierung, 14 Bits für die Byte-Selektierung, 2 ignoriert. So ist es nötig, fünfmal über die Adreßleitungen Daten zu senden, damit die Adresse endgültig feststeht.

Der Sprachsynthesizer hat je zwei Ein- und Ausgabewege: a) den Kommandospeicher und b) den FIFO-Puffer für die Eingabe; c) den Datenregister und d) den

Statusregister zur Ausgabe von Daten an den Computer. Sie haben folgende Funktionen:

zu a) Bereitstellung der Befehle des CPU für den VSP

zu b) Speichern der Sprachdaten vom Benutzer. Der FIFO (so genannt, weil die ersten ankommenden Daten als erste den Puffer verlassen) ist die Wurzel allen Unverständnisses. Seine Besonderheit ist, daß die Daten des Benutzers zwar ordentlich abgelegt, aber seltsamerweise ‚rückwärts' herausgeschoben werden. Genauer gesagt ist es ein parallel-ein, umgekehrt-seriell-heraus-Puffer. Beispiel: Lege ich die Zahl 49 (binär 00110001) auf den Puffer, so kommt diese als 140 (binär 10001100) wieder heraus! So wird sie dann weiter verwendet. Desweiteren verfügt der Puffer über zwei Statusanzeigen (Buffer low: BL, Buffer empty: BE).
Sind im Puffer nur 8 von 16 speicherbaren Bytes, meldet der VSP BL=1. Ist der Puffer endgültig ausgelaufen, ist BE=1, was den sofortigen Abbruch der Sprachsynthese zur Folge hat.

zu c) Formulierung eines Bytes. Die gewünschten Daten fließen seriell (normal) in den Datenregister und werden anschließend parallel an das CPU gegeben. Zu erreichen sind diese Daten über die Adresse SPCHRD (SPCHRD= >9000, SPCHWT= >9400)

zu d) Status der Sprachausgabe, abrufbar über den Datenbus auf D0, D1, D2. Das bedeutet: MOVB §SPCHRD,R1; R1=>8000 bedeutet, daß Sprachausgabe im Gange ist. R1=>C000 heißt, daß der Puffer unterbelegt ist (=BL). R1=>2000 heißt, daß der Puffer leer und die Sprachausgabe zu Ende ist.
Nun zu den Befehlen des VSP. Es gibt nämlich noch ein paar mehr, als in bisherigen Anleitungen zu lesen war. Alle diese sind über SPCHWT an den Synthesizer zu geben:

- > 00: NOP (bringt nichts)
- > 10: Byte lesen
- > 20: NOP
- > 30: Adresse lesen und einsetzen
- > 4X: Teil der Adresse schreiben (Nybble X)
- > 50: Aussprechen
- > 60: Aussprechen von externen Daten
- > 70: RESET

ZU DEN EINZELNEN BEFEHLEN:

– NOP heißt No OPeration, und das tut es auch.
– Byte lesen: über SPCHWT an den Prozessor gegeben, erhält man nach 12 Mikrosekunden auf SPCHRD ein Byte von einer Position im VSM.
– Adresse lesen und einsetzen: Zwei Bytes im VSM an der aktuellen Position werden gelesen und als neue Adresse für weitere Aktionen betrachtet.
– Adresse schreiben: Dieser Befehl wird fünfmal wiederholt, um eine neue Adresse im VSM zu setzen. Erlaubt gegenüber >30 auch einen Sprung zu parallelen VSMs.
– Aussprechen: Die Daten, die der aktuellen Adresse im VSM folgen, werden als Sprachdaten behandelt und ausgesprochen. Sobald ein Stopcode auftaucht oder der RESET ausgeführt wird, wird die Sprachausgabe unterbrochen.
– dito mit externen Daten: Hier wird der berüchtigte FIFO benutzt. Der VSP wartet, bis der Puffer halb voll ist und beginnt mit der Sprachausgabe. Alle Daten über SPCHWT kommen auf den FIFO und werden nicht als Befehle erkannt. Nur ein Stopcode kann den Sprachvorgang beenden.
– RESET: Setzt Synthesizer auf Urzustand zurück, ist aber nicht während der Ausgabe externer Daten wirksam.

WIE FUNKTIONIERT NUN DIE SPRACHSYNTHESE?

Die Daten werden entweder aus dem FIFO oder aus dem VSM geholt, auf Lautstärke, Filterkoeffizienten, Stimmhaftigkeit, Stimmlosigkeit usw. analysiert und über verschiedene Wege über einen Signalgenerator auf den D/A-Wandler und zum Lautsprecher gegeben. Am interessantesten ist in diesem Zusammenhang die Sprachcodierung. Ein Filter setzt sich aus mehreren Abschnitten zusammen. Diese werden von Bits dargestellt und nahtlos aneinandergehängt. Wir haben hier:

4 Bits Energie
1 Bit Wiederholung
6 Bits Tonhöhe
10mal 3–5 Bits Filterkoeffizienten, die den Klang bestimmen.

Von den Koeffizienten gibt es zehn, von denen die ersten beiden 5 Bits, die nächsten fünf 4 Bits und die letzten drei je 3 Bits benötigen. Leider kann ich zur Zeit nicht sagen, welche Bedeutung den einzelnen Koeffizienten zukommt. Zusammen ergeben sie 50 Bits in einem Filter bei 40 Filter pro Sekunde. Man hat jedoch Wege gefunden, die Datenmenge drastisch zu reduzieren:

Vokalische Laute verändern sich sehr langsam und erlauben Wiederholungen. Der Filter endet hier mit der Tonhöhe.
Stimmlose Laute brauchen dagegen nur vier Koeffizienten. Ihre Tonhöhe ist natürlich 0.
Kurze Pausen können erzeugt werden, wenn die Energie 0 ist. Hier endet der Rahmen. In einem Schaubild verdeutlicht sieht dies so aus:

Energie	WH	Tonhöhe	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	
//////	0	//////	//////	//////	//////	//////	//////	//////	//////	//////	//////	//////	komplett
//////	1	//////											Wiederholung
//////	0	0	//////	//////	//////	//////							stimmlos
0													Pause
15													Stopcode
4	1	6	5	5	4	4	4	4	4	3	3	3	Bits

Aufgabe der Programme ist nun, Filter zu dekodieren und zu erstellen.
Mit der Dekodierung muß man zunächst untersuchen, wie Laute gebildet werden, um mit dem zweiten Programm diese umzuändern. Die Programme sind in Extended Basic geschrieben, um nahezu allen Besitzern eines Synthesizers die Möglichkeit zu geben,

die Programme zu benutzen.

ERKLÄRUNGENZUM DEKODIER-PROGRAMM

Gültige Druckernamen sind PIO, RS232 u.a. Wenn kein Drucker vorhanden ist, einfach ENTER drücken. Ist ein Filter dargestellt, wartet der Computer auf Druck der ENTER-Taste, bevor er den nächsten Filter dekodiert.

ERKLÄRUNGEN ZUM KODIERER:

Der Computer wehrt automatisch zu große Werte ab. Achten Sie darauf, den String nicht zu groß werden zu lassen. Er darf 252 Bytes nicht überschreiten. Damit man ihn abspeichern kann, sollte er sogar nicht über 250 Bytes groß werden. Ist kein Drucker oder Disklaufwerk vorhanden, kann der Benutzer den String in seiner DATA-Form abschreiben. Das Rollen wird durch anhaltenden Tastendruck gestoppt.
So kann der String wieder gebildet werden:

```
10  DATA...(Daten vom
    Konverter)
20  DATA...(usw. bis
    alle Daten genannt)
100 READ A,B,C::
    A$=CHR$(A)&
    CHR$(B)&CHR$(C)
110 FOR X=1 TO C ::
    READ BYTE ::
    A$=A$&CHR$
    (BYTE) :: NEXT X
120 CALL SAY(,A$)
    ::END
```

Ansonsten dürfte alles klar sein. Wer weitere Informationen (besonders technische) über den Sprachsynthesizer haben möchte, möge mich bitte anschreiben; meine Adresse steht im Kopf der Programme.

```
1 !*********************
2 !* PROGRAMM ZUR ER- *
3 !* STELLUNG NEUER   *
4 !* WOERTER FUER DEN *
5 !* SPRACHSYNTHESIZER*
6 !*********************
7 ! 15.11.86   AUTOR:
8 ! MICHAEL ZAPF, HOMBURGER
STR.2, 6367 KARBEN 1
9 !
1Ø DISPLAY AT(8,1)BEEP ERASE
 ALL:"ENERGIE=": :"WIEDERHOL
UNG=": :"TONHOEHE= ": :"BYTES
 BELEGT="
2Ø DISPLAY AT(18,1):"1  2  3
  4  5  6  7  8 9 1Ø":RPT$("
-",27)
3Ø ACCEPT AT(8,1Ø)SIZE(2)VAL
IDATE(DIGIT):EN :: IF EN>15
THEN 3Ø
4Ø IF EN=15 THEN ENDFLG=1
5Ø CALL ENCODE(4,ENDFLG,SP$,
(EN))
6Ø IF EN=Ø THEN 3Ø ELSE IF E
NDFLG THEN 17Ø
7Ø ACCEPT AT(1Ø,15)SIZE(1)VA
LIDATE("Ø1"):RP
8Ø CALL ENCODE(1,Ø,SP$,(RP))
9Ø ACCEPT AT(12,11)SIZE(2)VA
LIDATE(DIGIT):PITCH :: IF PI
TCH>63 THEN 9Ø
1ØØ CALL ENCODE(6,Ø,SP$,(PIT
CH))
11Ø IF RP=1 THEN 3Ø ELSE IF
PITCH=Ø THEN ENDE=4 ELSE END
E=1Ø
12Ø CALL HCHAR(2Ø,1,32,32)::
 FOR A=1 TO ENDE :: IF A<8 T
HEN SPALTE=1+(A-1)*3 :: ANZ=
4 ELSE SPALTE=6+A*2 :: ANZ=3
13Ø IF A<3 THEN ANZ=5
14Ø ACCEPT AT(2Ø,SPALTE)VALI
DATE(DIGIT)SIZE(2):K :: IF K
>2^ANZ-1 THEN 14Ø
15Ø CALL ENCODE(ANZ,Ø,SP$,K)
16Ø NEXT A :: GOTO 3Ø
17Ø CALL CLEAR :: IF LEN(SP$
)>252 THEN PRINT "STRING IST
 ZU LANG!" :: STOP ELSE SP$=
" '"&CHR$(Ø)&CHR$(LEN(SP$))&S
P$
18Ø DISPLAY AT(1,1)BEEP ERAS
E ALL:SP$
19Ø DISPLAY AT(1Ø,1):"1->DRU
CKEN ALS DATALISTE":"2->STRI
NG AUF DISK SAVEN":"3->DATAS
 AUF BILDSCHIRM":"4->AUSSPRE
CHEN":"5->ENDE"
2ØØ CALL KEY(Ø,K,S):: IF K<4
9 QR K>53 THEN 2ØØ :: ST=Ø
21Ø ON K-48 GOTO 22Ø,24Ø,26Ø
,31Ø,32Ø
22Ø INPUT "DRUCKERNAME:":P$
:: IF P$="" THEN 18Ø ELSE OP
EN #1:P$
23Ø FOR A=1 TO LEN(SP$):: PR
INT #1,USING "### ":ASC(SEG$
(SP$,A,1));:: NEXT A :: CLOS
E #1 :: GOTO 18Ø
24Ø IF LEN(SP$)>253 THEN PRI
NT "ZU LANG FUER EINEN DATEN
SATZ" :: INPUT "<ENTER>":EN$
 :: GOTO 18Ø
25Ø INPUT "FILENAME? ":F$ ::
 IF F$="" THEN 18Ø ELSE OPEN
 #1:F$,INTERNAL,OUTPUT,VARIA
BLE 254 :: PRINT #1:SP$ :: C
LOSE #1 :: GOTO 18Ø
26Ø FOR A=1 TO 7 :: ST=ST+1
:: PRINT USING "### ":ASC(SE
G$(SP$,ST,1));
27Ø IF ST=LEN(SP$)THEN 3ØØ
28Ø NEXT A
29Ø CALL KEY(Ø,K,S):: IF K<>
-1 THEN 29Ø :: GOTO 26Ø
3ØØ INPUT "<ENTER>":ENT$ ::
GOTO 18Ø
```

```
310 CALL SAY(,SP$):: GOTO 18
0
320 END
330 SUB ENCODE(ANZ,ENDFLG,SP
$,X)
340 GRENZE=2^ANZ :: IF FACT=
0 THEN FACT=1
350 FOR Y=1 TO ANZ
360 X1=X*2 :: X=X1+GRENZE*(X
1>(GRENZE-1)):: BITPOS=(BITP
OS+1)AND 7
370 VALUE=VALUE-(X<>X1)*FACT
 :: FACT=FACT*2
380 IF BITPOS=0 THEN SP$=SP$
&CHR$(VALUE):: PLATZ=PLATZ+1
 :: FACT=1 :: VALUE=0
390 NEXT Y :: IF ENDFLG AND
BITPOS THEN SP$=SP$&CHR$(VAL
UE):: PLATZ=PLATZ+1
400 DISPLAY AT(14,14):PLATZ+
3
410 SUBEND
```

```
1 !*********************
2 !* DEKODIERPROGRAMM  *
3 !* FUER DAS VOKABU-  *
4 !* LAR DES  SPRACH-  *
5 !* SYNTHESIZERS      *
6 !*********************
7 !13.11.86 AUTOR:
8 !MICHAEL ZAPF, HOMBURGER S
TR.2, 6367 KARBEN 1
9 !
10 IMAGE "    ##  "
20 IMAGE "  # "
30 IMAGE "    ##     "
40 IMAGE " ## "
50 CALL CLEAR :: INPUT "DRUC
KERNAME:":P$ :: IF P$<>"" TH
EN OPEN #1:P$
60 INPUT "WELCHER AUSDRUCK?
":A$ :: CALL SPGET(A$,SPGET$
):: CALL SAY(,SPGET$)
70 SP$=SEG$(SPGET$,4,LEN(SPG
ET$)-3)
80 PLATZ=1
90 DISPLAY AT(24,5):"<ENTER>
" :: ACCEPT AT(24,12):ENT$
100 Z$="WORT="&CHR$(34)&A$&C
HR$(34):: DISPLAY AT(1,1)BEE
P ERASE ALL:Z$ :: IF P$<>""
THEN PRINT #1:Z$: : :
110 DISPLAY AT(8,1):"ENERGIE
=": :"WIEDERHOLUNG=": :"TONH
OEHE="
120 DISPLAY AT(18,1):"1  2
3  4  5  6  7  8 9 10":RPT$(
"-",27)
130 IF P$<>"" THEN PRINT #1:
"ENERGIE WH TONHOEHE  K1  K2
  K3  K4  K5  K6  K7  K8  K9
  K10":RPT$("-",60)
140 CALL HCHAR(8,11,32,4)::
CALL HCHAR(10,16,32,2):: CAL
L HCHAR(12,12,32,4):: CALL H
CHAR(20,3,32,26)
150 CALL DECODE(4,PLATZ,SP$,
EN):: DISPLAY AT(8,9):EN
160 IF P$<>"" THEN PRINT #1,
USING 10:EN;
170 IF EN=15 THEN STOP ELSE
IF EN=0 THEN CALL WAIT(P$)::
 GOTO 140
180 CALL DECODE(1,PLATZ,SP$,
X):: DISPLAY AT(10,14):X
190 IF P$<>"" THEN PRINT #1,
USING 20:X;
200 CALL DECODE(6,PLATZ,SP$,
PITCH):: DISPLAY AT(12,10):P
ITCH
210 IF P$<>"" THEN PRINT #1,
USING 30:PITCH;
220 IF X=1 THEN CALL WAIT(P$
):: GOTO 140
230 IF PITCH=0 THEN MK=4 ELS
E MK=10
240 FOR K=1 TO MK :: IF K<3
THEN ANZ=5 ELSE IF K>7 THEN
ANZ=3 ELSE ANZ=4
250 CALL DECODE(ANZ,PLATZ,SP
$,X)
260 IF K<8 THEN DISPLAY AT(2
0,1+(K-1)*3):STR$(X)ELSE DIS
PLAY AT(20,6+K*2):STR$(X)
270 IF P$<>"" THEN PRINT #1,
USING 40:X;
280 NEXT K :: CALL WAIT(P$):
: GOTO 140
290 SUB DECODE(ANZ,PLATZ,SP$
,X)
300 X=0 :: FACT=2^(ANZ-1)
310 FOR Y=1 TO ANZ :: IF BIT
POS=0 THEN VALUE=ASC(SEG$(SP
$,PLATZ,1)):: PLATZ=PLATZ+1
320 V1=VALUE/2 :: VALUE=INT(
V1):: BITPOS=(BITPOS+1)AND 7
330 X=X-(V1<>VALUE)*FACT ::
FACT=FACT/2
340 NEXT Y
350 SUBEND
360 SUB WAIT(P$): DISPLAY A
T(24,10):"<ENTER>" :: ACCEPT
 AT(24,17):ENT$ :: CALL HCHA
R(24,1,32,32)
370 IF P$<>"" THEN PRINT #1
380 SUBEND
```

TI99/4A

PREISSENKUNG BEI VIELEN ARTIKELN !!

Peripherie

RS 232 Karte (Atronic)	249,–
32 K-Karte (Atronic)	249,–
Compact Peripherie System CPS 99 mit 1 Diskettenlaufwerk DSDD + Statistikmodul	999,–
CPS 99 mit 2 Diskettenlaufwerken DSDD + Statistikmodul	1329,–
Diskettenlaufwerk DSDD mit Einbausatz für Periph.box	349,–
2 Diskettenlaufwerke DSDD mit Einbau- und Kabelsatz für P.box	729,–
Externe 32 K-Erweiterung batteriegepuffert	149,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle	249,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle + Druckerkabel + Epsondrucker LX 86	1009,–
dto. + Epsondrucker FX 800	1379,–
dto. + Epsondrucker EX 800	1679,–
dto. + Stardrucker NL-10	1079,–
dto. + Stardrucker NG-10	1149,–
80-Zeichen-Karte (Mechatronic)	449,–
Ext. V-24-Schnittstelle (2 x RS 232) mit druchgeschl. Bus	199,–
MBx-Sprachsteuereinheit + Baseballmodul anschlußfertig	299,–
WIPO Mouse99	199,–
RGB-Modulator mit Scartstecker	179,–
Akustikkoppler Dataphon S 21 d + externe V-24-Schnittstelle + Verbindungskabel	499,–
Akustikkoppler Dataphon S 21 d + Kabel für RS 232 Karte	279,–
Joystickinterface + 2 Joysticks Quickshot II	85,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter)	115,–
Modulexpander 3-fach	125,–

Modulsoftware

Extended Basic II Plus	229,–
Mini Memory	169,–
Mini Memory+ TMS 9900 Assemblerhandbuch Mini Memory dt.	225,–
Editor/Assembler (32 K notw.)	159,–
TI-Writer (32 K notw.)	199,–
Multiplan (32 K notw.)	199,–
TI-Logo II (32 K notw.)	199,–
Diskfixer (Navarone)	109,–
Terminal Emulator II	85,–
Connect four, Attack	je 19,–
Alpiner, Chisholm Trail, Dthello, Car Wars, Tomstone City	je 29,–
Defender, Dig Dug, Invaders, Munchman, Moonmine, Statistik je	39,–
Fathom, Jungle Hunt, Moon Patrol, Hopper, Parsec, Indoore Soccer	je 49,–
Congo Bongo, Burgertime, Espial, Moonsweeper, Microsurgeon, Treasure Island, Donkey Kong, Tunnels of doom, Protector II, Shamus	je 59,–
Adenvturemodul, Buck Rogers, Return to Pirates Isle, Star Trek, Video Chess, Popeye	je 69,–
Pole Position, Miner 2049	je 79,–
TI-Artist (Diskette)	99,–
Turbo Pascal (Diskette)	149,–

Bücher

Editor/Assembler Handbuch dt.	98,–
TMS 9900 Assemblerhandbuch für das Mini Memory dt.	78,–
TI-99/4 A Intern dt.	38,–
Assembler-Kurs (99er-Journal)	59,–
Introduction to Assembly Language (englsich)	49,–
Grafiktablett Supersketch + Deferndеr + Dig Dug+ Donkey Kong + Statistik	199,–

Commodore/ Schneider

Commodore 64-II + VC 1541	875,–
Commodore SX-64	1479,–
Commodore C 128 D	1169,–
Amiga mit Farbmonitor 1081	2479,–
Schn. CPC 6128 Grünmonitor	889,–
Schn. CPC 6128 Farbmonitor	1499,–
Schneider PC mit Schwarzweißmonitor und 1 Laufwerk	1849,–
dto. + 2 Laufwerke	2249,–
Schneider PC mit Farbmonitor und 1Laufwerk	2249,–
dto. + 2 Laufwerke	2679,–

Versandkostenpauschale (Warenwert bis DM 1000,–/darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–). Lieferung nur gegen NN oder Vorauskasse; Ausland nur Vorauskasse. Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstraße 5, 7324 Rechberghausen, Tel. (07161) 5 28 89

DISKONT-RECHNUNG

Das Programm „Diskontrechnung" dient dazu, die Gebühren für einen normalen oder bundesbankfähigen Wechsel festzustellen, dabei werden die allgemein gültigen Diskontvereinbarungen der Banken und der Bundesbank beachtet.

Nach dem Vorspann erreichen Sie das Menü. Hier können Sie sich entscheiden, ob Sie einen normalen Wechsel (normaler Handelswechsel) oder einen Bundesbankwechsel diskontieren wollen.

Beim normalen Wechsel müssen Sie zunächst den Wechselbetr ag eingeben. Danach können Sie wählen, ob Sie Diskont- und Verfalldatum oder die Laufzeit in Tagen angeben wollen. Außerdem wird der Diskontsatz der Bundesbank benötigt und der Gebührensatz der jeweiligen Bank (z.B. Volksbank z.Zt. 6,– DM). Im weiteren fragt Sie der Computer, ob der Wechsel schon versteuert wurde, ob der Wechsel auf einen Bankplatz zahlbar gestellt ist (d.h. auf eine Stadt, an der sich eine Landeszentralbank oder die Bundesbank befindet). Als letztes können Sie noch den Mehrwertsteuersatz angeben, falls der Steuersatz 14% beträgt, können Sie einfach mit ENTER weitergehen.

Auf dem Bildschirm erscheint nun die Aufstellung aller Abzüge (z.B. Diskont, Steuer) und den Auszahlungsbetrag. Auf Wunsch wird Ihnen auch eine komplette Aufstellung aller Spesen + MwSt. vermittelt, die Sie direkt so als Spesenabrechnung für den Schuldner übernehmen können.

Falls Sie einen bundesbankfä higen Wechsel diskontieren wollen, müssen Sie nur Punkt 2 des Menüs wählen, den Wechselbetrag, die Diskontlaufzeit und den Diskontsatz angeben. Der Computer erstellt auch hier eine vollständige Aufstellung aller Spesen Das Programm erleichtert das Berechnen der Wechselspesen allgemein, was vor allem das Disponieren eines Wechsel einfacher macht.

IMPRESSUM

HCA (MSX&TI-REVUE, die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.

Geschäftsführer: Werner E. Seibt

Redaktion: Alfons Bauer, Thomas Dehn, Jürgen Puschmann, Uwe Schröder

Verantwortlich für den Inhalt:
MSX-Teil: Werner E. Seibt
TI-Teil: Oliver Huber (Technik) Hartmut König (Listings)

Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d
Alle: Postach 1107, 8044 Lohhof.

Anfragen bitte nur schriftlich

Druck: Dierichs, Kassel

Es gilt die Honorarliste des Verlages.

Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.

Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.

Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

©by CA Verlag, Heßstraße 90, 8000 München 40
SPS und Autoren

Erstverkaufstag: 11.3.1987

```
10 ! ***********************
11 ! *                     *
12 ! *   DISKONTRECHNUNG   *
13 ! *                     *
14 ! *    Copyright by     *
15 ! *                     *
16 ! *    Jochen Zeier     *
17 ! *                     *
18 ! * Benoetigte Geraete  *
19 ! *  TI99/4A  Konsole   *
20 ! *    Ext. Basic       *
21 ! *                     *
22 ! *  Speicherbelegung   *
23 ! *    4612 Bytes       *
24 ! *                     *
25 ! ***********************
26 !
9Ø ! BILDSCHIRMVORBEREITUNG
11Ø CALL CLEAR :: CALL SCREE
N(5):: FOR I=Ø TO 14 :: CALL
 COLOR(I,16,1):: NEXT I
12Ø ! VORSPANN
13Ø DISPLAY AT(5,8):"DISKONT
RECHNUNG";TAB(8);"==========
=====" 
14Ø DISPLAY AT(1Ø,13):"VON"
15Ø DISPLAY AT(14,1):"
  -------------
  -HAMSTISOFT-
 --------------
 -JOCHEN ZEIER-"
16Ø DISPLAY AT(18,8):"------
--------"
17Ø DISPLAY AT(24,1):">IRGEN
DEINE TASTE DRUECKEN<"
18Ø CALL KEY(Ø,K,S):: IF S T
HEN 2ØØ ELSE 18Ø
19Ø ! MENUE
2ØØ CALL CLEAR
21Ø DISPLAY AT(1,1Ø):"M E N
U E"TAB(1Ø);"=========="
22Ø DISPLAY AT(6,1):"1...NOR
M.WECHSELDISKONTIER."
23Ø DISPLAY AT(8,1):"2...BUN
DESBANKDISKONTIERUNG"
24Ø DISPLAY AT(1Ø,1):"3...EN
DE"
25Ø DISPLAY AT(14,5):"IHRE W
AHL =>1" :: ACCEPT AT(14,17)
BEEP SIZE(-1)VALIDATE("123")
:WAHL
26Ø ON WAHL GOTO 3ØØ,76Ø,98Ø
27Ø !
28Ø ! NORMALE WECHSEL-
      DISKONTIERUNG
29Ø !
3ØØ CALL CLEAR
31Ø DISPLAY AT(1,1):"NORMALE
 WECHSELDISKONTIERUNG=======
====================="
32Ø DISPLAY AT(5,1):"WECHSEL
BETR.= ØØØØØØØØ.ØØ DM" :: CA
LL E(5,8,WB)
33Ø CALL ZEIT(T)
34Ø DISPLAY AT(11,1):"DISKON
TSATZ =        ØØ.ØØ %" :: CA
LL E(11,2,DS)
35Ø DISPLAY AT(14,1):"GEBUEH
REN    =       ØØ.ØØ DM" :: C
ALL E(14,2,G)
36Ø DISPLAY AT(16,1):"IST DE
R WECHSEL VERSTEUERT:J" :: A
CCEPT AT(16,28)BEEP SIZE(-1)
VALIDATE("JN"):W1$
37Ø DISPLAY AT(18,1):"IST DE
R WECHSEL ZAHLBAR AUF EINEN
BANKPLATZ GESTELLT  :J" :: A
CCEPT AT(19,28)BEEP SIZE(-1)
VALIDATE("JN"):W2$
38Ø DISPLAY AT(21,1):"MWST-S
ATZ    =        14.ØØ %" :: AC
CEPT AT(21,21)SIZE(-2)VALIDA
TE(DIGIT):MS
39Ø ! BERECHNUNG
4ØØ IF T<5 THEN T=5
41Ø D=INT((WB*T*DS/36ØØØ)*1Ø
Ø+.5)/1ØØ :: IF D<2 THEN D=2
42Ø IF W1$="J" THEN WS=Ø ELS
E WS=(INT(WB/1ØØ)+1)*.15
43Ø IF W2$="J" THEN IG=Ø ELS
E IG=INT((D/1ØØØ)*1ØØ+.5)/1Ø
Ø :: IF IG<5 THEN IG=5
44Ø GA=D+WS+G+IG
45Ø BW=WB-GA
46Ø MW=INT((GA/1ØØ*MS)*1ØØ+.
5)/1ØØ
47Ø RE=GA+MW
48Ø ! AUSGABE
49Ø CALL HCHAR(5,1,32,64Ø)
5ØØ DISPLAY AT(5,1):USING "W
ECHSELBETR.= ########.## DM"
:WB
51Ø DISPLAY AT(8,1):USING ".
/. DISKONT = #### ####.## DM"
:D
52Ø DISPLAY AT(1Ø,1):USING "
./. W-STEU. = # ###### ##.## DM
":WS
53Ø DISPLAY AT(12,1):USING "
./. GEBUEHR =         ##.## DM
":G
54Ø DISPLAY AT(14,1):USING "
./. IK-GEB. = ########.## DM
":IG
55Ø DISPLAY AT(16,15):"=====
========="
56Ø DISPLAY AT(18,1):USING "
AUSZAHLUNG   = ########.## DM
":BW
57Ø DISPLAY AT(2Ø,15):"=====
```

LISTINGS

```
=========="
580 DISPLAY AT(24,1):"ERSTEL
.D.SPESENABRECHNUNG :J" :: A
CCEPT AT(24,28)BEEP SIZE(-1)
VALIDATE("JN"):WAHL$
590 IF WAHL$="N" THEN 200
600 CALL CLEAR
610 DISPLAY AT(1,7):"SPESENA
BRECHNUNG";TAB(7);"=========
=======" 
620 DISPLAY AT(6,1):USING "D
ISKONT      = ########.## DM"
:D
630 DISPLAY AT(8,1):USING "W
ECHSELSTEU.= ########.## DM"
:WS
640 DISPLAY AT(10,1):USING "
GEBUEHR     =       ##.## DM
":G
650 DISPLAY AT(12,1):USING "
IK-GEBUEHR  = ########.## DM
":IG
660 DISPLAY AT(14,15):"-----
---------"
670 DISPLAY AT(16,15):USING
" ###### #.## DM":GA
680 DISPLAY AT(18,1):USING "
## % MWST.  = ########.## DM
":MS,MW
690 DISPLAY AT(20,15):"=====
========="
700 DISPLAY AT(22,15):USING
"########.## DM":RE
710 DISPLAY AT(24,15):"=====
========="
720 CALL KEY(0,K,S):: IF S T
HEN 200 ELSE 720
730 !
740 ! BUNDESBANK-
      DISKONTIERUNG
750 !
760 CALL CLEAR
770 DISPLAY AT(1,1):"  BUNDE
SBANKDISKONTIERUNG      =====
=================="
780 DISPLAY AT(5,1):"WECHSEL
BETR.= 00000000.00 DM" :: CA
LL E(5,8,WB)
790 CALL ZEIT(T)
800 DISPLAY AT(11,1):"DISKON
TSATZ =      00.00 %" :: CA
LL E(11,2,DS)
810 ! BERECHNUNG
820 IF T<5 THEN T=5
830 IF T<11 THEN G=2 ELSE G=
0
840 D=INT((WB*T*DS/36000)*10
0+.5)/100
850 BW=WB-D-G
860 ! AUSGABE
870 CALL HCHAR(5,1,32,640)
880 DISPLAY AT(5,1):USING "W
ECHSELBETR.= ########.## DM"
:WB
890 DISPLAY AT(8,1):USING ".
/. DISKONT = ########.## DM"
:D
900 DISPLAY AT(11,1):USING "
./. GEBUEHR =         #.## DM
":G
910 DISPLAY AT(13,15):"=====
========="
920 DISPLAY AT(15,1): USING "
AUSZAHLUNG  = ########.## DM
":BW
930 DISPLAY AT(17,15):"=====
========="
940 CALL KEY(0,K,S): IF S T
HEN 200 ELSE 940
950 !
960 ! ENDE
970 !
980 CALL CLEAR :: END
990 !
1000 ! SUBPROGRAMME
1010 !
1020 SUB E(ZEI,LG,EDAT)
1030 ACCEPT AT(ZEI,23-LG)BEE
P SIZE(-LG)VALIDATE(DIGIT):G
E
1040 ACCEPT AT(ZEI,24)SIZE(-
2)VALIDATE(DIGIT):KE
1050 EDAT=GE+KE/100
1060 SUBEND
1070 !
1080 SUB ZEIT(T)
1090 DISPLAY AT(8,1):"WELCHE
 DATEN SIND BEKANNT:
                      1...D
ISKONT- U. VERFALLTAG  2...Z
EITRAUM IN TAGE"
1100 DISPLAY AT(13,5):"IHRE
WAHL =>1" :: ACCEPT AT(13,17
)SIZE(-1)VALIDATE("12"):WAHL
1110 CALL HCHAR(8,1,32,192):
: ON WAHL GOTO 1120,1140
1120 DISPLAY AT(8,1):"DISKON
TTAG  =    00.00.00  VERFAL
LTAG  =    00.00.00" :: CALL
 DZE(T)
1130 IF T<0 THEN 1120 ELSE S
UBEXIT
1140 DISPLAY AT(8,1):"DISKON
TTAGE =   0000.00 T." :: A
CCEPT AT(8,19)SIZE(-4)VALIDA
TE(DIGIT):T
1150 SUBEND
1160 !
1170 SUB DZE(ZEI)
1180 ACCEPT AT(8,18)SIZE(-2)
VALIDATE(DIGIT):TAG :: IF TA
```

```
G=Ø OR TAG>31 THEN 118Ø ELSE
 IF TAG=31 THEN TAG=3Ø
119Ø ACCEPT AT(8,21)SIZE(-2)
VALIDATE(DIGIT):MON :: IF MO
N=Ø OR MON>12 THEN 119Ø
12ØØ ACCEPT AT(8,24)SIZE(-2)
VALIDATE(DIGIT):JAH
121Ø ACCEPT AT(9,18)SIZE(-2)
VALIDATE(DIGIT):VTAG :: IF V
TAG=Ø OR VTAG>31 THEN 121Ø E
LSE IF VTAG=31 THEN VTAG=3Ø
122Ø ACCEPT AT(9,21)SIZE(-2)
VALIDATE(DIGIT):VMON :: IF V
MON=Ø OR VMON>12 THEN 122Ø
123Ø ACCEPT AT(9,24)SIZE(-2)
VALIDATE(DIGIT):VJAH
124Ø ZEI=(VJAH-JAH)*36Ø :: Z
EI=ZEI+(VMON-MON)*3Ø :: ZEI=
ZEI+VTAG-TAG
125Ø SUBEND
```

STAR WRITER

Ich habe schon viele Briefprogramme aus Computer-Zeitschriften ausprobiert, von denen gesagt wurde, sie seien für Textverarbeitung geeignet. Diese Behauptung sollte jedoch offenbar nur dem Umsatz des Heftes dienen. Es konnten nur Briefe einfachster Art geschrieben werden. Da keines dieser Programme meinen Ansprüchen genügte, stellte ich aus Tests von professionellen Textverarbeitungsprogrammen jene Anforderungen zusammen, die mein Programm erfülllen sollte. Bei der Programmierung stieß ich an die Grenzen des auf dem TI in XBasic machbaren. Ich achtete dabei auf anwenderfreundliches Handling. Das Programm ermöglicht Ihnen z.B.: verschiedene Brieftypen (mit und ohne Adressfenster, mit und ohne Absenderangabe, Rundschreiben); Wahl der Schriftart (Pica oder Elite); Fettschrift, Unterstreichung auch nur eines einzelnen Wortes; Zentrieren eines Satzes, mehrere Leerzeilen; Einfügen, Kopieren oder Verschieben eines Satzes; Suchen eines bestimmten Wortes innerhalb des Textes; Ausdruck eines Adress-Etiketts. Das Ganze wird auch noch übersichtlich durch drei Statuszeilen mit den wichtigsten Informationen über den Text. Und nun der Clou: Das alles benötigt weder Speichererweiterung noch Diskettenlaufwerk.

Anleitung:

1> Eingabe

Zuerst fragt Sie der Computer, ab welchem Block Sie eingeben wollen. Hier haben Sie die Möglichkeit, entweder in einem begonnenen Text fortzufahren (wenn Block>5 ist) oder einen neuen Text zu beginnen. Nun können Sie zwischen Pica-(Normal-) und Eliteschrift wählen. Jetzt bestimmen Sie den Tabulator und die Anzahl der Zeichen/Zeile. Achten Sie darauf, daß die Summe aus Tab. und Z./Zeile nicht >80 ist. Am bequemsten ist es, wenn Sie die vorgegebenen Werte übernehmen, die ein sauberes Bild garantieren. Darauf folgt die Wahl des Brieftypes:

a) Normalbrief. Hier wird beim Ausdruck der Absender (= Block 1 – 3) und das Datum(= Block 4) angegeben. Die Texteingabe erfolgt ab Block 5.

b) Fensterbrief. Zusätzlich werden die Empfängerangaben an einem für Fensterbriefumschlägen passenden Feld ausgedruckt. Die entsprechenden Angaben (Datum, Anrede, Name, Straße, PLZ/Ort, Land) werden bei der Eingabe abgefragt. Damit das Adressfeld genau in das Brieffenster paßt, sollte der Tabulator nicht verändert werden.

c) Text ohne Kopf: die Texteingabe beginnt bei Block 1. Absender- und Empfängerangabe entfallen.

d) Rundschreiben (mit Fenster): wie Fensterbrief; Block 5 – 9 bleiben frei. Dadurch wird beim Ausdruck der Platz für die Empfängeradresse freigehalten, um später einen Adressaufkleber anzubringen.

Regeln für die Eingabe:

– Sie können den Text einer Zeile bis vor das blaue '<'-Zeichen eingeben
– Bestätigen Sie die Eingabe mit 'Enter'. Sie können statt 'Enter' auch FCTN X (eine Zeile nach unten) oder FCTN E (eine Zeile nach oben – etwas länger drücken!) zum Blättern im Text verwenden.
– Wenn Sie 'Eingabe' oder 'Ändern' verlassen wollen, drücken Sie am Anfang einer Zeile (auch wenn die Zeile schon beschrieben ist –>, diese bleibt dann trotzdem erhalten) CTRL E und 'Enter'.
– Achtung: Wollen Sie einen Block ändern, so müssen Sie einmal mit dem Cursor auf das blaue '<'-Zeichen gehen. Dann erst darf man die Zeile ändern.
– Ein freier Block gilt beim Ausdruck als Leerzeile.
– Sie können Steuerzeichen verwenden. Diese werden als kleine, blaue, umrahmte Buchstaben auf dem Bildschirm dargestellt. Sie werden beim Ausdruck natürlich nicht mitgedruckt (sondern der gewünschte Effekt).

Steuerzeichen gelten beim Ausdruck nicht als Leerzeichen. Man kann also auch innerhalb eines Wortes mehrere verschiedene Steuerzeichen verwenden.
Es können mehrere Steuerzeichen hintereinandergeschrieben werden (z.B. fzu). Nun die Steuerzeichen im einzelnen:

Fettdruck: (f) erzeugt durch CTRL F. Dieses Steuerzeichen wird, wenn Eliteschrift gewählt wurde, beim Ausdruck ignoriert (da hier nicht möglich).
Unterstreichung: (u) erzeugt durch CTRL G.
Neue Seite: (n)= CTRL H: nur am Anfang einer Zeile eingeben! Keine anderen Steuerzeichen verwenden! Nach (n) die Seiten-Nr. eingeben. Beim Ausdruck werden Sie dann aufgefordert, ein neues Blatt einzulegen. Wenn Sie 'Enter' gedrückt haben, schreibt der Drucker die Seiten-Nr. zentriert zwischen zwei Bindestrichen aus. Danach kommen automatisch mehrere Leerzeilen.
Zentrieren: (z = CTRL J: ebenfalls nur am Anfang einer Zeile eingeben. Nur (l) ist als Steuerzeichen vor dem (z) erlaubt. (z) zentriert die ganze Zeile beim Ausdruck. Hinter (z) gelten wieder die normalen Eingaberegeln.
Steuerzeichen ausschalten: (s) = CTRL K: damit werden alle in einem Block vorher verwendeten Steuerzeichen (f) und (u) beendet. Bei (l), (z) und (n) ist dies nicht nötig.
Leerzeile: (l)= CTRL L: Schreibt eine Leerzeile beim Ausdruck. (l) kann beliebig oft hintereinander geschrieben werden, jedoch nur am Anfang einer Zeile! Danach kann normal mit dem Text fortgefahren werden.
Übrigens: Deutsche Sonderzeichen erreichen Sie mit:
Ä = FCTN R/Ü= FCTN T/Ö= VCTN Z
ä =FCTN F/ü= FCTN G/ö= FCTN A/ß= FCTN W
Ein DIN-A 4-Fensterbrief hat im 1. Blatt mit ein-Zeilen-Vorschub max. 45 Blocks, ein DIN-A 4-Normal-

Bitte lesen Sie weiter auf Seite 54

LISTINGS

```
1 ! ****************************
2 ! * S T A R W R I T E R *
3 ! *      COPYRIGHT BY        *
4 ! *   HEINRICH HAMBACH       *
5 ! *    ALIEN SOFTWARE        *
6 ! *                          *
7 ! * BENOETIGTE GERAETE:      *
8 ! *  TI99/4A  KONSOLE        *
9 ! *      EXT. BASIC          *
10 !* EPSON-KOMP. DRUCKER       *
11 !*  KASSETTENREKORDER        *
12 !*  SPEICHERBELEGUNG:        *
13 !*       6924 BYTES          *
14 !*************************
15 !
16 ! ACHTUNG: ZEILEN 1 - 18
17 !   WEGEN SPEICHERMANGEL
18 !     NICHT EINGEBEN!!
100 ZZ=70 :: E$=CHR$(27):: D
IM A$(72),K$(73):: DEF T$=RP
T$(" ",T-1): N$=CHR$(10):
DEF L$(K)=RPT$(N$,K)
110 CALL CLEAR :: CALL C ::
FOR I=1 TO 12 :: CALL COLOR(
I,2,15):: NEXT I :: CALL SCR
EEN(6):: CALL CHAR(128,"FF80
9E90909E80FF"):: CALL COLOR(
0,15,6,13,6,15,14,6,15)
120 ON WARNING NEXT :: DISPL
AY AT(9,5)BEEP:"S t a r W r
i t e r": : : :"     "&CHR$(1
28)&" BY ALIEN SOFTWARE"
130 CALL A(0,0,A$(),0):: RES
TORE :: FOR I=97 TO 114 :: R
EAD C$ :: CALL CHAR(I,C$)::
NEXT I :: FOR I=1 TO 15 :: R
EAD N,C$ :: CALL CHAR(N,C$):
: NEXT I
140 DATA 0000003804 3C443A,00
4040407844443 8,000000003C40404
03C,0004040 43C44443A,0000003
844784 03C
150 DATA 0018242 07C202020,00
00003C443C0438,0040407844444
444,001000181010103C,0008000
80808 4830
160 DATA 0040405060504848,00
101010101012 0C,000000B649494
949,0000005864444444,0000003
C44444438
170 DATA 0000003844784040,00
000038443C0404,0000005824202
020
180 DATA 116,002020702020201
8,117,0000004444444 43A,121,0
000004444 3C0438,126,00182428
24242820
190 DATA 123,0044003804 3C443
A,124,0044003C44444438,125,0
0440044444443A
200 DATA 91,42001824427E4242
,92,42003C424242423C,93,4200
424242424 23C,133,FF818199A5B
9A19D
210 DATA 134,FF899591BD91919
1,135,FF8181A5A5A5A599FF8181
B9A5A5A5A5,138,FF8181BD8991A
1BDFF81819DA19985B9FF81A1A1A
1A1A599,64,0010101000101010
220 P$="MEN]" :: GOSUB 870 :
: DISPLAY AT(7,5):"1= >TEXT
EINGEBEN": :"     2= >EDITIER
EN": :"     3= >TEXT LESEN":
:"     4= >TEXT DRUCKEN"
230 DISPLAY AT(15,5):"5= >AD
RESSETIKETT": :"     6= >AUF
KASSETTE": :"     7= >VON KAS
SETTE": :"     8= >TEXT L\SCH
EN": :"     9= >ENDE"
240 CALL KEY(0,K,S):: IF K<4
9 OR K>57 THEN 240 ELSE CALL
 C :: ON K-48 GOTO 250,430,5
60,590,590,810,820,830,850
250 P$="EING." :: GOSUB 880
:: DISPLAY AT(10,1):">1 TEXT
 EINGEBEN": :"EINGABE AB B
LOCK NR.";BL+1 :: ACCEPT AT(
13,22)SIZE(-3)BEEP:Z
260 IF Z>5 THEN 360 ELSE DIS
PLAY AT(13,1):"W[HLEN SIE UN
TER FOLGENDEN  SCHRIFTARTEN:
 1": :"1> PICA-SCHRIFT":"2>
ELITE-SCHRIFT"
270 ACCEPT AT(14,15)SIZE(-1)
VALIDATE("12")BEEP:W :: IF W
=1 THEN M$="PICA" :: T=7 ::
EL$="" ELSE M$="ELITE" :: T=
8 :: EL$=E$&"M"
280 ZZ=79-T :: GOSUB 880
290 DISPLAY AT(13,1):"TABULA
TOR      :";T: :"ZEICHEN/ZEIL
E :";ZZ: : : :: ACCEPT AT(13
,17)SIZE(-2)BEEP:T :: ACCEPT
 AT(15,17)SIZE(-2)BEEP:ZZ ::
 IF ZZ+T>80 THEN GOSUB 860 :
: GOTO 290
300 GOSUB 880 :: DISPLAY AT(
13,1)BEEP:"1 NORMALBRIEF": :
"2 FENSTERBRIEFUMSCHLAG": :"
3 TEXT OHNE KOPF": :"4 RUNDS
CHREIBEN(MIT FENSTER)"
310 CALL KEY(0,K,S):: W=K-48
 :: IF S=0 OR W<1 OR W>4 THE
N 310 ELSE IF W=1 THEN B$="N
ORM." :: D=5 ELSE IF W=2 THE
N B$="FENS." :: D=10 ELSE IF
 W=3 THEN B$="O.KO." :: D=1
:: GOTO 350 ELSE IF W=4 THEN
 B$="RUND." :: D=10
320 BL=1 :: A$(1)="HEINRICH
HAMBACH" :: A$(2)="REGENSBUR
```

LISTINGS

```
GER STR. 236" :: A$(3)="
8500 NJRNBERG 30"
330 GOSUB 870 :: DISPLAY AT(
6,1):"DATUM  :" :: ACCEPT AT
(6,10)BEEP:A$(4):: IF D=5 OR
 B$="RUND." THEN 350
340 DISPLAY AT(9,1):"ANREDE
:": : :"NAME   :": : :"STR.
 :": : :"PLZ/ORT:": : :"LAN
D   :" :: FOR I=1 TO 5 :: CA
LL A(6+I*3,10,A$(),I+4):: NE
XT I
350 BL=D-1 :: Z=D
360 GOSUB 870 :: W=5
370 DISPLAY AT(W,1):"Z."&STR
$(Z):">>"&A$(Z)&RPT$(" ",ZZ-
LEN(A$(Z))):: CALL SPRITE(#W
,60,5,(W+INT((3+ZZ)/28))*8,(
(4+ZZ)-28*INT((3+ZZ)/28))*8)
380 IF K=200 THEN RETURN
390 CALL A(W+1,3,A$(),0):: K
=LEN(A$(0)):: S=LEN(A$(Z))::
 IF K>ZZ THEN GOSUB 860 :: G
OTO 390 ELSE IF SEG$(A$(0),1
,1)=CHR$(133)THEN 220 ELSE I
F K>25 OR((K=8 OR S=0)AND A$
(Z)<>A$(0))THEN A$(Z)=A$(0)
400 BL=MAX(Z,BL):: W=W+4 ::
Z=Z+1 :: CALL KEY(0,K,S):: I
F K=11 THEN W=W-8 :: Z=Z-2 :
: IF W<5 THEN W=21
410 IF W>21 THEN W=5 :: GOSU
B 880
420 IF Z<1 OR Z>72 THEN 220
ELSE 370
430 P$="EDIT." :: GOSUB 880
:: DISPLAY AT(8,1):">2 TEXT
EDITIEREN": : :"1) BL[TTERN
U. [NDERN ": :"2) EINF]GEN E
INES BLOCKS"
440 DISPLAY AT(15,1):"3) VER
SCHIEBEN EINES BLOCKS": :"4)
 KOPIEREN EINES BLOCKS": :"5
) SUCHEN EINES WORTES" :: CA
LL KEY(0,K,S):: IF S=0 THEN
440 ELSE CALL C :: ON K-48 G
OTO 450,510,470,490,530
450 DISPLAY AT(11,1):"[NDERN
 AB BLOCK NR." :: ACCEPT AT(
11,21)BEEP:Z :: GOTO 360
460 K=200 :: Z=W :: W=19 ::
A$(Z)="" :: GOSUB 370 :: K=0
 :: CALL A(W+1,3,A$(),Z):: G
OTO 220
470 DISPLAY AT(11,1):"WELCHE
R BLOCK SOLL VERSCHO- BEN WE
RDEN?" :: ACCEPT AT(12,13)BE
EP:S :: GOTO 510
480 A$(W)=A$(S):: A$(S)="" :
: GOTO 220
490 DISPLAY AT(11,1):"WELCHE
R BLOCK SOLL KOPIERT  WERDEN
?" :: ACCEPT AT(12,9)BEEP:S
:: GOTO 510
500 A$(W)=A$(S):: GOTO 220
510 DISPLAY AT(14,1):"HINTER
 WELCHER ZEILE SOLL   DIESER
 BLOCK EINGESETZT     WERDEN
? " :: ACCEPT AT(16,9)BEEP:W
:: W=W+1
520 FOR I=W TO BL :: K$(I+1)
=A$(I):: NEXT I :: FOR I=W T
O BL+1 :: A$(I)=K$(I):: K$(I
)="" :: NEXT I :: Z=Z+1 :: B
L=BL+1 :: IF K=52 THEN 500 E
LSE IF K=51 THEN 480 ELSE 46
0
530 W=0 :: A=9 :: DISPLAY AT
(6,1):"WORT":"BEFINDET SICH
IN DEN ZEILEN:" :: ACCEPT AT
(6,6):P$ :: FOR I=1 TO 72 ::
 IF POS(A$(I),P$,1)=0 THEN 5
50
540 DISPLAY AT(A,1+W):STR$(I
)&"," :: W=W+3 :: IF W>26 TH
EN W=0 :: A=A+1
550 NEXT I :: ACCEPT BEEP:P$
 :: GOTO 220
560 W=5 :: P$="LESEN" :: GOS
UB 880 :: FOR I=1 TO 72 :: P
$=STR$(I)&".  "&A$(I):: DISP
LAY AT(W,1):P$&RPT$(" ",ZZ+5
-LEN(A$(I)))
570 W=W+4 :: IF W>21 THEN W=
5 :: DISPLAY AT(24,1):"WEITE
R=ENTER / MEN]=E" :: ACCEPT
AT(24,27):P$ :: IF P$<>"" TH
EN 220
580 NEXT I :: INPUT ">> WEIT
ER MIT ENTER ":P$ :: GOTO 22
0
590 OPEN #1:"PIO" :: IF K=53
 AND B$="FENS." THEN 650 ELS
E IF K=53 THEN 800
600 P$="DRUCK" :: GOSUB 880
:: DISPLAY AT(10,1):">5 TEXT
 DRUCKEN": : :"(DRUCKEN BEEN
DEN MIT 'E' !)"
610 DISPLAY AT(15,1):"DRUCKE
N VON BLOCK 1":"      BIS
BLOCK";BL :: ACCEPT AT(15,19
)SIZE(-2):B :: ACCEPT AT(16,
19)SIZE(-2):Y
620 DISPLAY AT(18,1):"VORSCH
UB: 1  ZEILE(N)": :"ANZAHL
AUSDRUCKE: 1" :: ACCEPT AT(1
8,11)SIZE(-3):W :: ACCEPT AT
(20,19)SIZE(-2):AN
630 PRINT #1:E$;"A";CHR$(12*
W);E$;"8";:: FOR I=1 TO AN :
: F=B :: IF I>1 THEN GOSUB 7
```

LISTINGS

```
2Ø
64Ø IF B>=1 AND D=1 OR B>4 T
HEN 67Ø ELSE F=D :: W=ZZ+T-L
EN(A$(3)&A$(4))-2 :: PRINT #
1:EL$:T$&A$(1);TAB(W);A$(2):
N$:TAB(W);A$(3)&", "&A$(4):L
$(6):: IF D=5 THEN 67Ø
65Ø PRINT #1:CHR$(15);TAB(12
);A$(1)&"*"&A$(2)&"*"&A$(3);
CHR$(18);L$(2);EL$:T$&A$(5):
N$:T$&A$(6):N$:T$&A$(7):L$(2
):T$&A$(8):N$:T$&A$(9):L$(4)
66Ø IF K=53 THEN 8ØØ ELSE PR
INT #1:"":L$(3)
67Ø FOR A=F TO Y :: DISPLAY
AT(21,1):STR$(A)&" >"&A$(A)&
RPT$(" ",ZZ+6-LEN(A$(A))):
CALL KEY(Ø,W,S):: IF W=1Ø1 O
R W=69 THEN 8ØØ
68Ø F=T :: FOR W=134 TO 14Ø
:: IF POS(A$(A),CHR$(W),1)>Ø
 THEN 7ØØ
69Ø NEXT W :: PRINT #1:EL$;T
$&A$(A):: GOTO 79Ø
7ØØ PRINT #1:EL$;T$;:: FOR W
=1 TO LEN(A$(A)):: C$=SEG$(A
$(A),W,1)
71Ø IF C$<>CHR$(136)THEN 74Ø
72Ø DISPLAY AT(21,1)BEEP:"*
BITTE NEUES BLATT EINLEGEN
DANN ENTER DRJCKEN" :: CALL
KEY(Ø,K,S):: IF K<>13 THEN 7
2Ø ELSE IF C$<>CHR$(136)THEN
 RETURN
73Ø PRINT #1:N$:CHR$(13);TAB
(T+(ZZ-LEN(A$(A))+3)/2);"- "
;SEG$(A$(A),2,ZZ);" -":L$(3)
:: GOTO 79Ø
74Ø IF ASC(C$)<134 THEN 77Ø
ELSE IF C$=CHR$(134)AND EL$=
"" THEN C$="E" ELSE IF C$=CH
R$(135)THEN C$="-"&CHR$(1)
75Ø IF C$=CHR$(138)THEN F=T+
(ZZ-LEN(A$(A))/2 ELSE IF C$
=CHR$(139)THEN C$="F"&E$&"-"
&CHR$(Ø)ELSE IF C$=CHR$(14Ø)
THEN PRINT #1:N$:T$;:: GOTO
78Ø
76Ø PRINT #1:TAB(F);E$;C$;::
 GOTO 78Ø
77Ø PRINT #1:TAB(F);C$;:: F=
F+1
78Ø NEXT W
79Ø PRINT #1:N$:E$;"F";E$;"-
";CHR$(Ø);:: NEXT A :: NEXT
I
8ØØ CLOSE #1 :: GOTO 22Ø
81Ø OPEN #1:"CS1",INTERNAL,S
EQUENTIAL,OUTPUT,FIXED 192 :
: PRINT #1:M$,EL$,ZZ,T,B$,BL
,D :: FOR I=1 TO BL STEP 2 :
: DISPLAY AT(15,1):"BLOCK #"
&STR$(I):: PRINT #1:A$(I),A$
(I+1):: NEXT I :: GOTO 8ØØ
82Ø OPEN #1:"CS1",INTERNAL,S
EQUENTIAL,INPUT ,FIXED 192 :
: INPUT #1:M$,EL$,ZZ,T,B$,BL
,D :: FOR I=1 TO BL STEP 2 :
: DISPLAY AT(15,1):"BLOCK #"
&STR$(I):: INPUT #1:A$(I),A$
(I+1):: NEXT I :: GOTO 8ØØ
83Ø DISPLAY AT(1Ø,1):">8 TEX
T L\SCHEN": :"BLOCK 1  BIS
 BLOCK 72 WERDEN": :"NUN GEL
\SCHT !" :: ACCEPT AT(13,7)S
IZE(-2):K :: ACCEPT AT(13,2Ø
)SIZE(-2):S :: IF K<Ø OR K>7
2 OR S<Ø OR S>72 THEN 83Ø
84Ø FOR I=K TO S :: A$(I)="
" :: NEXT I :: GOTO 22Ø
85Ø BREAK :: GOTO 11Ø
86Ø CALL SOUND(2ØØ,22Ø,Ø)::
DISPLAY AT(22,1):"":"* LINE
TOO LONG. TRY AGAIN:":"" ::
RETURN
87Ø CALL C
88Ø DISPLAY AT(1,1):"@TAB
 : ";T;TAB(15);"@BRIEF :";B$
:"@Z/ZEILE: ";ZZ;TAB(15);"@B
LOCKS:";BL;"@MODUS  :";M$;TA
B(15);"@PROGR.:";P$:RPT$("-"
,28)
89Ø CALL VCHAR(1,3Ø,64,3)::
RETURN
9ØØ SUB A(Z,S,A$(),N):: IF Z
>Ø THEN 92Ø
91Ø FOR I=32 TO 64 :: B$=B$&
CHR$(I):: NEXT I :: FOR I=91
 TO 143 :: B$=B$&CHR$(I):: N
EXT I :: SUBEXIT
92Ø ACCEPT AT(Z,S)SIZE(-8Ø)V
ALIDATE(UALPHA,B$)BEEP:A$(N+
Ø):: SUBEND :: SUB C :: CALL
 DELSPRITE(ALL):: FOR I=5 TO
 24 :: DISPLAY AT(I,1):: NEX
T I :: CALL VCHAR(1,1,31,48)
:: CALL VCHAR(1,31, 31,48)::
SUBEND
```

STARWRITER

Fortsetzung von Seite 51

brief im 1. Blatt max. 55 Blocks und ein Text ohne Kopf pro DIN-A 4-Blatt max. 60 Blocks!

2> Ändern

Hier er scheint wieder ein Menü:
1. Blättern und Ändern: Sie werden gefragt, ab welcher Zeile Sie ändern und blättern wollen. Dann gelten die gleichen Regeln wie bei 'Regeln für die Eingabe'.

Bei Punkt 2, 3, 4 werden Sie gefragt, hinter welcher Zeile der Block eingesetzt, kopiert bzw. verschoben

werden soll. Achtung: Wenn Sie keine Speichererweiterung besitzen, dürfen Sie diese 3 Punkte nur anwählen, wenn das Schriftstück <50 Blocks ist.
5. Suchen nach einem Wort: Hier können Sie das zu suchende Wort oder den zu suchenden Satz eingeben. Nun schreibt der Computer jeden Block heraus, in dem das gewünschte Wort/Satz vorkommt.

3> Text lesen

Dieses Unterprogramm erklärt sich von selbst.

4> Text drucken

Sie haben die Möglichkeit, den Beginn und das Ende Ihres Briefes selbst zu bestimmen, damit Sie auch einzelne Ausschnitte Ihres Textes ausdrucken lassen können. Dann fragt Sie der Computer nach Zeilenvorschub (auch Kommastellen, z.B. 1.5) und die Anzahl der Ausdrucke *z.B. für Rundschreiben oder Kopie für Ihre Unterlagen). Bei Normalbrief, Fensterbrief und Rundschreiben wird die Absenderangabe mit Datum ausgegeben. Beim Fensterbrief schreibt der Drucker die Absenderadresse nochmals in Kleinschrift im Fensterausschnitt. Beim Rundschreiben wird der Fensterausschnitt mit Leerzeilen ausgegeben. Darauf kann dann ein Adress-Etikett (z.B. von einem Dateiprogramm) angebracht werden. Um das richtige Falten des Briefes (mit Fensterausschnitt) zu erleichtern, wird an einer vorgesehenen Stelle am linken Rand ein Punkt gedruckt.

5> Adress-Etikett

Dieses Unterprogramm druckt einen Adress-Aufkleber aus. Dies ist aber nur möglich, wenn vorher ‘Fensterbrief‘ gewählt wurde. Der Grund für diese Option ist, daß für Pakete und Päckchen ein sauberes und ansprechendes Empfänger-Etikett gewünscht wird.

6> auf Kassette und 7> von Kassette

Hier können Sie Ihren geschriebenen Text speichern bzw. laden.

8> Text löschen

Sie werden gefragt, welche Block-Abschnitte gelöscht werden sollen. Falls Sie diesen Punkt versehentlich gewählt haben, so geben Sie einfach 0 und 0 ein.

9> Ende

Wenn Sie diesen Punkt gewählt haben, können Sie, wenn Sie im Programm selbst nichts geändert haben, mit ‘Con‘ das Programm wieder starten, wobei alle eingegebenen Daten erhalten bleiben.

Hinweise zum Programm selbst:

Statuszeilen: zeigt alle wichtigen Parameter an (eingestellter Tab., Anzahl der möglichen Zeichen/Zeile, Schriftart (Pica oder Elite), Briefart, Anzahl der belegten Blocks, Unterprogramm (z.B. Menü, Eingabe usw.)).

In Zeile 320 geben Sie bitte Ihre eigene Adresse an. In Zeile 590 geben Sie Ihren eigenen Drucker-Eröffnungsbefehl an. Bei einer RS-232-Schnittstelle nehmen Sie am besten die größtmögliche Übertragungsrate. Wenn Sie eine Speichererweiterung besitzen, können Sie überall im Programm die Zahl 72(= höchstmögliche Blockanzahl) entsprechend Ihrem Bedarf abändern.

```
100 !**********************
110 !*   MASKENGENERATOR  *
120 !*                    *
130 !*    Copyright  by   *
140 !*    RUDOLF KARY     *
150 !*                    *
160 !* Benoetigte Geraete *
170 !*  TI-99/4A Konsole  *
180 !*     EXT. BASIC     *
190 !*    KASS.REC. CS1   *
200 !*                    *
210 !**********************
230 DIM MA$(100,4)
240 CALL CLEAR
250 CALL SCREEN(12)
260 FD=1 !! Filedescriptor f
uer Tape
270 DISPLAY AT(10,5):"Masken
generator"
280 DISPLAY AT(12,1):"Aufbau
 eines Datensatzes zumAnlege
n einer neuen Datei"
290 DISPLAY AT(14,1):"mit de
m Dateiverwaltungs-   progra
mm"
300 DISPLAY AT(24,1):"Weiter
 mit Enter!"
310 ACCEPT H$
320 !------------------
330 ! Masken generieren
340 !------------------
350 CALL CLEAR
360 CALL SCREEN(4)
370 MS=28 !max. Spalten
380 DATF=0 ! Anzahl der Nett
odatenfelder in der Maske.
390 Z=1 ! Zeiger auf Abbild
Array MA$(,)
400 DISPLAY AT(12,1)BEEP:"Bi
tte geben Sie einen Namen fu
er die neue Datei an?"
410 ACCEPT AT(24,1):MA$(0,0)
420 ST=1 ! Zeilenabstand
430 ENDZ=23 ! letzte Zeile
440 DISPLAY AT(6,1)ERASE ALL
:"Sollen die variablen Daten
  des Satzes mit ____ markie
rt werden?(J/N)"
450 DISPLAY AT(12,1):"Beispi
el mit Unterstrich:    Vorn
ame:______ "
460 DISPLAY AT(14,1):"Beispi
el ohne Unterstrich:   Vorn
ame:       "
470 DISPLAY AT(17,1):"Es ist
 die Eingabelaenge im Fall o
hne nicht erkennbar."
480 DISPLAY AT(19,1):"Dafuer
 koennen mehr Daten   unterg
ebracht werden da fuerden Un
terstr. 1 zeile reser-"
490 DISPLAY AT(22,1):"viert
wird."
```

```
500 DISPLAY AT(24,12):"(J/N)
?:"
510 ACCEPT AT(24,20)VALIDATE
("J","N"):H$
520 IF H$="J" THEN ENDZ=ENDZ
-1 :: ST=ST+1
530 DISPLAY AT(2,1)ERASE ALL
:"Es ist jetzt jeder Variabl
enein Name zuzuordnen.Die La
ge am Bs ist beliebig."
540 DISPLAY AT(5,1):"dann is
t die Laenge des Ein-gabefel
des mit ...  zu kenn-zeichne
n. Die Laenge kann"
550 DISPLAY AT(8,1):"Null se
in dann kann aber in einer D
atei mit diesem Satz-layout
kein Wert zugewiesen werden"
560 DISPLAY AT(24,1):"Weiter
 mit Enter!"
570 ACCEPT H$
580 DISPLAY AT(1,1)ERASE ALL
:MA$(0,0)
590 !
600 FOR I=2 TO ENDZ STEP ST
610 S=1 ! Spaltenzaehler auf
 Zeilenanfang
620 DISPLAY AT(24,1):"Name d
er Satzvariablen?"
630 ACCEPT AT(I,S)SIZE(MS-S+
1)BEEP:MA$(Z,0)
640 H=LEN(MA$(Z,0))
650 IF H=0 THEN 810
660 !
670 MA$(Z,1)=STR$(I)
680 MA$(Z,2)=STR$(S)
690 !
700 S=S+H
710 IF S>=MS THEN 810
720 DISPLAY AT(24,1):" '.' m
arkiert Eingabefeld"
730 ACCEPT AT(I,S)SIZE(MS-S+
1)VALIDATE(".")BEEP:HS$
740 H=LEN(HS$)
750 IF H<>0 THEN DATF=DATF+1
760 MA$(Z,3)=STR$(S)
770 MA$(Z,4)=STR$(H)
780 Z=Z+1 !naechster Eintrag
 in MA$
790 S=S+H
800 IF MS-S>2 THEN 620
810 NEXT I
820 Z=Z-1
830 MA$(0,1)=STR$(Z)
840 MA$(0,2)=STR$(ST)!Kennun
g mit/ohne Markierung
850 MA$(0,3)="0" ! noch kein
e Datei vorhanden
860 MA$(0,4)=STR$(DATF)!Anza
hl Nettodatenfelder in Maske
nabbild eintreagen.
870 CALL MAUF(MA$(,)! Test:
Maske am BS aufbauen
880 DISPLAY AT(24,1):"Maske
auf CS1 speichern?"
890 ACCEPT AT(24,26)VALIDATE
("J","N"):H$
900 IF H$="N" THEN GOTO 950
910 DISPLAY AT(21,4)ERASE AL
L:"Kassette in CS1 zum siche
rn der Datensatzbeschreibung
!"
920 OPEN #FD:"CS1",OUTPUT,IN
TERNAL,FIXED 192
930 CALL WRITE(FD,MA$(,),Z,4
)! speichern des Maskenabbil
des
940 CLOSE #FD :: CALL CLEAR
950 DISPLAY AT(12,4)ERASE AL
L:"Maskengenerator Ende"
960 END
970 !---------------
980 ! Maske aufbauen
990 !-------------
1000 SUB MAUF(MA$(,))
1010 DISPLAY AT(1,1)ERASE AL
L:MA$(0,0)
1020 H=VAL(MA$(0,1))
1030 IF H=0 THEN BREAK ! es
kann keine Maske aufgebaut w
erden: FEHLER
1040 CALL CHAR(128,"0038")
1050 FOR I=1 TO H
1060 Z=VAL(MA$(I,1))
1070 S=VAL(MA$(I,2))
1080 SN=VAL(MA$(I,3))
1090 DISPLAY AT(Z,S)SIZE(SN-
S):MA$(I,0)
1100 IF VAL(MA$(0,2))=2 THEN
 CALL HCHAR(Z+1,SN+2,128,VAL
(MA$(I,4)))!DF markieren
1110 NEXT I :: SUBEND
1120 !--------------------
1130 !komprimiert speichern
1140!--------------------
1150 SUB WRITE(FD,ARRAY$(,),
XMAX,YMAX)
1160 PRINT #FD:XMAX,YMAX ::
ANZAHL=0
1170 FOR X=0 TO XMAX :: FOR
Y=0 TO YMAX
1180 LAEN=LEN(ARRAY$(X,Y)+1
1190 ANZAHL=ANZAHL+LAEN
1200 IF ANZAHL<190 THEN 1220
1210 PRINT #FD:"@" :: ANZAHL
=LAEN
1220 PRINT #FD:ARRAY$(X,Y),
1230 NEXT Y :: NEXT X
1240 PRINT #FD:"@"
1250 SUBEND
```

++ 99-SOFT präsentiert:+++ PAINTER, das Malprogramm der Spitzenklasse. Mit über 25 Funktionen wie FILL, DRAW, COPY, CIRCLE. Läuft auf jedem TI mit 32 K und Disk. Insgesamt 28 K; 100 % M-code. Keine Raubkopie! Nur 40 DM inkl. Porto etc. Günther Rehm, T. 08261/1208

TI REVUE

Das Magazin für TI 99-4A

Nr. 3/87

Einführung in GPL (IV)

WICHTIGER HINWEIS!
Ein Hinweis für die TI-REVUE-Abonnenten! Dies ist die Nummer 3/87 der TI-REVUE!, aber die Nummer 5 der HCA-MSX-REVUE. Und so wird auch bei uns in der Abo-Verwaltung gezählt!

Im Test: XB-Extender

Dialog

Sechs Seiten Börse

DAS HEFT IM HEFT FÜR TI-USER

mispa Microcomputer
bietet an

ASSEMBLER SEMINAR TI 99

vom 01.05.87 bis 03.05.87 veranstalten wir einen Kurs, der Sie in die Grundlagen und Geheimnisse der Maschinenprogrammierung des TI-99/4A einführt.

Fordern Sie nähere Unterlagen kostenlos bei uns an:

mispa Microcomputer, Roland Reinl
Münchner Str.48, 8025 Unterhaching
Telefon 089 / 61 90 48

Grüß Gott -

Mit der letzten Ausgabe fanden zwei bekannte Zeitschriften ein neues Gewand. Als Zeichen des Interesses und der aktiven Mitarbeit unserer Leser wurden wir mit positiven, aber auch negativen Meinungen überschüttet. Sehr vielen Lesern fiel ein Stein vom Herzen, daß endlich eine akzeptable Lösung gefunden wurde, zwei hervorragende Computersysteme weiterhin aktiv mit Rat und Tat zu unterstützen, nachdem diese beiden Rechner von der gesamten übrigen Fachpresse seit einiger Zeit totgeschwiegen werden.

Wir haben lange überlegt. Von Auflagenreduzierung bis zur völligen Einstellung der TI-REVUE glitten unsere Gedanken. Doch treue Leser, die jahrelang Freud und Leid mit uns teilten, kann man nicht einfach vor den Kopf stoßen und in einem Satz erklären: „Es ist aus, wir wollen nicht mehr." Und ob wir wollen. Es mußte nur ein Weg gefunden werden, die beständig kleiner werdende Schar der TI-User wenigstens kostendeckend zufriedenstellen zu können. Die Umstellung der TI-REVUE auf reinen Abonnement-Service war nicht der richtige Weg, wie wir an der Reaktion unserer Leser feststellen konnten. Man vermißte allgemein die Präsenz in den Zeitschriftenläden und am Kiosk und zu allem Überfluß deklassierte uns die Druckerei aufgrund dieser Tatsache zur Zweitklassigkeit und ließ nicht nur uns, sondern auch unsere Abonnenten unverhältnismäßig lange warten. Nun ist sie wieder voll da, die TI-REVUE.

Allerdings im Rahmen einer Fusion zweier Zeitschriften. Für alle der beste Weg, wie wir finden. Keine der beiden Anwendergruppen hat darunter gelitten. Nach unseren Erfahrungen profitieren einige Leser sogar doppelt an dem neuen Konzept. Nicht wenige freundeten sich mit dem MSX an, ohne den altgeliebten TI99 dadurch in der Abstellkammer verschwinden zu lassen. Diese Gruppe bezieht nun 2 Hefte zum Preis eines einzigen, ohne auch nur auf eine einzige Seite verzichten zu müssen und sie können zusätzlich unseren Leserservice gleich zweimal in der Woche wie gewohnt nutzen — Dienstags für ihren TI99 und Freitags für ihren MSX-Rechner.

Und unser Vorteil? Wir haben damit zu Gunsten der Leser unsere Druckkosten gesenkt. Nicht mehr und nicht weniger.

Bis zum nächsten Heft,
Ihr HCA-Team

IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Geschäftsführer: Werner E. Seibt
Redaktion: Alfons Bauer, Thomas Dehn, Jürgen Puschmann, Uwe Schröder
Verantwortlich für den Inhalt:
MSX-Teil: Werner E. Seibt
TI-Teil: Oliver Huber (Technik) Hartmut König (Listings)
Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d
Alle: Postach 1107, 8044 Lohhof.
Anfragen bitte nur schriftlich
Druck: Dierichs, Kassel
Es gilt die Honorarliste des Verlages.
Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.
Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

©by CA Verlag, Heßstraße 90, 8000 München 40
SPS und Autoren

Erstverkaufstag: 22.4.1987

TI99/4A

PREISSENKUNG BEI VIELEN ARTIKELN !!

Peripherie

RS 232 Karte (Atronic)	249,–
32 K-Karte (Atronic)	249,–
Compact Peripherie System CPS 99 mit 1 Diskettenlaufwerk DSDD + Statistikmodul	999,–
CPS 99 mit 2 Diskettenlaufwerken DSDD + Statistikmodul	1329,–
Diskettenlaufwerk DSDD mit Einbausatz für Periph.box	349,–
2 Diskettenlaufwerke DSDD mit Einbau- und Kabelsatz für P.box	729,–
Externe 32 K-Erweiterung batteriegepuffert	149,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle	249,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle + Druckerkabel + Epsondrucker LX 86	1009,–
dto. + Epsondrucker FX 800	1379,–
dto. + Epsondrucker EX 800	1679,–
dto. + Stardrucker NL-10	1079,–
dto. + Stardrucker NG-10	1149,–
80-Zeichen-Karte (Mechatronic)	449,–
Ext. V-24-Schnittstelle (2 x RS 232) mit druchgeschl. Bus	199,–
MBx-Sprachsteuereinheit + Baseballmodul anschlußfertig	299,–
WIPO Mouse 99	199,–
RGB-Modulator mit Scartstecker	179,–
Akustikkoppler Dataphon S 21 d + externe V-24-Schnittstelle + Verbindungskabel	499,–
Akustikkoppler Dataphon S 21d + Kabel für RS 232 Karte	279,–
Joystickinterface + 2 Joysticks Quickshot II	85,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter)	115,–
Modulexpander 3-fach	125,–

Modulsoftware

Extended Basic II Plus	229,–
Mini Memory	169,–
Mini Memory + TMS 9900 Assemblerhandbuch Mini Memory dt.	225,–
Editor/Assembler (32 K notw.)	159,–
TI-Writer (32 K notw.)	199,–
Multiplan (32 K notw.)	199,–
TI-Logo II (32 K notw.)	199,–
Diskfixer (Navarone)	109,–
Terminal Emulator II	85,–
Connect four, Attack	je 19,–
Alpiner, Chisholm Trail, Othello, Car Wars, Tomstone City	je 29,–
Defender, Dig Dug, Invaders, Munchman, Moonmine, Statistik je	39,–
Fathom, Jungle Hunt, Moon Patrol, Hopper, Parsec, Indoore Soccer	je 49,–
Congo Bongo, Burgertime, Espial, Moonsweeper, Microsurgeon, Treasure Island, Donkey Kong, Tunnels of doom, Protector II, Shamus	je 59,–
Adenvturemodul, Buck Rogers, Return to Pirates Isle, Star Trek, Video Chess, Popeye	je 69,–
Pole Position, Miner 2049	je 79,–
TI-Artist (Diskette)	99,–
Turbo Pascal (Diskette)	149,–

Bücher

Editor/Assembler Handbuch dt.	98,–
TMS 9900 Assemblerhandbuch für das Mini Memory dt.	78,–
TI-99/4 A Intern dt.	38,–
Assembler-Kurs (99er-Journal)	59,–
Introduction to Assembly Language (englsich)	49,–
Grafiktablett Supersketch + Defernder + Dig Dug + Donkey Kong + Statistik	199,–

Commodore / Schneider

Commodore 64-II + VC 1541	875,–
Commodore SX-64	1479,–
Commodore C 128 D	1169,–
Amiga mit Farbmonitor 1081	2479,–
Schn. CPC 6128 Grünmonitor	889,–
Schn. CPC 6128 Farbmonitor	1499,–
Schneider PC mit Schwarzweißmonitor und 1 Laufwerk	1849,–
dto. + 2 Laufwerke	2249,–
Schneider PC mit Farbmonitor und 1 Laufwerk	2249,–
dto. + 2 Laufwerke	2679,–

Versandkostenpauschale (Warenwert bis DM 1000,–/darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–).
Lieferung nur gegen NN oder Vorauskasse; Ausland nur Vorauskasse.
Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstraße 5, 7324 Rechberghausen, Tel. (07161) 5 28 89

DEFEKT AN DER PIO

Ich besitze das MBI Centronics Interface. Während des Ablaufs eines meiner Programme gab die Schnittstelle ihren Geist auf. Es wurde nicht mehr der Text gedruckt, sondern erst kurz vor Ende des Programms einige willkürliche Zeichen. Auch im Direktmodus wurden die Druckerbefehle nicht mehr ausgeführt. Der Drucker funktioniert im Testmodus einwandfrei. Kann es sein, daß die Schnittstellensoftware durch einen Programmfehler eliminiert wurde oder liegt ein anderer Defekt vor? Das Einstecken der Schnittstelle stellte nämlich immer ein Problem dar.
Johann Brescher, Bürmos

Eine Zerstörung der Software in der Schnittstelle ist aus einem Programm heraus nicht möglich. Es kann jedoch sein, daß Sie die Schnittstelle im eingeschaltetem Zustand anschlossen. Es ist daher ratsam, die Schnittstelle einzuschicken, da eine Fehlersuche sehr mühsam ist. Es besteht noch die Möglichkeit, daß die Kontakte der Schnittstelle oder der Konsole verschmutzt sind. Reinigen Sie diese also zuerst und probieren Sie es aus. Funktioniert es dann immer noch nicht, ist der Defekt wahrscheinlich an der Schnittstelle zu suchen.

ID-DATA FEHLERHAFT?

Ich besitze das Programm ID-DATA. Da ich sehr viele Adressen zu verwalten habe, ist dieses Programm eine sehr gute Hilfe. Leider habe ich seit einiger Zeit ein Problem. Wenn ich wieder einmal Adressen eingeben möchte und diese sortiere, so bleibt das Programm nach ca. 110 Adressen hängen. Es schafft es nicht mehr, da angeblich zu wenig Speicherplatz zur Verfügung steht. Wo liegt das Problem?
M. Glanemann, Greyen

Sollte Ihr Programm wirklich „hängen“ bleiben und keine Eingabe mehr annehmen, ist es wirklich zerstört. Doch ich glaube, daß das Problem bei Ihnen an einer ganz anderen Stelle zu suchen ist. Ich gehe sicherlich nicht falsch in der Annahme, daß Sie als externes Speichermedium das Original-Laufwerk benutzen. Mit diesem Laufwerk können nur 90 KB Programme, bzw. Daten gespeichert werden. Bereits beim Anlegen einer neuen Datei wird von I.D-DATA auf der Diskette Platz für 500 Datensätze reserviert. Hinzu kommen noch das Index-Register, eine Datei mit den voreingestellten Parametern und eventuell eine Datei, um die erstellten Datensätze mit dem TI-Writer oder dem Editor des E/A-Moduls weiterbearbeiten zu können. Mehr als Dreiviertel der Diskette sind damit bereits belegt. Zum Sortieren von mehreren hundert Datensätzen reicht natürlich der Arbeitsspeicher nicht aus. Deshalb legt sich I.D-DATA für den Sortiervorgang unbemerkt ein Hilfs-File auf Diskette an, welches nach erfolgtem Sortierlauf sofort wieder gelöscht wird. Je umfangreicher die bereits eingegebenen Datenbestände, desto größer muß das Hilfs-File auf Diskette ausfallen. Und genau hier stößt das Datenverwaltungsprogramm an die Grenzen des Original-Laufwerks. Die 90 KB reichen dafür nicht aus. Ist ein zweites Laufwerk angeschlossen, wird dieses von ID-DATA zum Sortieren mitbenutzt. Im anderen Fall generiert das Programm bei Auslastung des Hilfs-Files die Meldung „Speicherplatz reicht für vollständiges Sortieren nicht aus“ und bricht den Sortiervorgang nach der Erstellung des neuen Index-Files ab. Hier kann man sich aber durchaus weiterhelfen, indem man den Sortierlauf mehrmals hintereinander startet. Am Ende ist es durchaus möglich, daß ein oder zwei Datensätze beim Sortieren übergangen wurden. Auch ein weiterer Sortierlauf hilft da nicht weiter. Machen Sie sich von den wenigen nicht sortierten Datensätzen mit der Taste 4 eine Kopie auf den Drucker, löschen die Sätze an dieser Stelle komplett und geben sie noch einmal neu ein. Nach dem nächsten Sortierlauf werden auch diese Sätze an der richtigen Stelle stehen. Zugegeben, dieser Vorgang ist sehr zeitintensiv, doch damit wurden bei mir bereits 500 angelegte Datensätze korrekt sortiert. Wer ein Laufwerk mit zwei Schreib/Leseköpfen oder zwei Laufwerke besitzt, hat übrigens mit diesen Problemen bei I.D-DATA nicht zu kämpfen.

WAS MACHE ICH MIT DEN SONDERTASTEN?

Ich komme mit den Steuertasten nicht zurecht. Vielleicht können Sie mir weiterhelfen, wie man die Steuertasten benutzt und wozu sie eigentlich nützlich sind?
Chr. Lefen, Krefeld

Die Grundfunktionen der sogenannten Steuertasten sind bereits im Handbuch zum TI99 unter „Spezielle Tastenfunktionen“ beschrieben. Diese Beschreibung erklärt jedoch nur, wie die Sondertasten zum Editieren von Programmen eingesetzt werden können. Sicherlich ist Ihnen im Zusammenhang mit den Sondertasten die Tastaturschablone im Lieferumfang Ihres TIs aufgefallen. Erwerben Sie nun zusätzlich ein anderes Modul, wie z.B. das Editor/Assembler oder den TI-Writer, bekommen Sie jeweils wieder andere Schablonen ausgehändigt. Das heißt, die Sondertasten werden von den Programmautoren umbelegt, um eine optimale Nutzung der Software zu gewährleisten. So können Sie bei den beiden angesprochenen Modulen zum Beispiel durch einen Tastendruck den Bildschirm in eine beliebige Richtung scrollen, von der Texteingabe zur Kommandoebene wechseln oder ganze Blöcke verschieben oder löschen. Der Möglichkeiten gibt es viele. Selbstverständlich können Sie diese Sondertasten auch für Ihre eigenen Programme umprogrammieren. Den ASCII-Code dieser Tasten finden Sie im Handbuch und in der Referenzkarte zum Extended-Basic- »

UNSER ☏ TELEFONSERVICE ☏

Wie immer steht unseren Lesern unser Telefon-Service zur Verfügung! Jeden Dienstag von 15 bis 19 Uhr. Listings/Programme: 0 89 /129 80 13

Modul. Auch durch ein kleines, selbstgeschriebenes Programm können Sie diesen Code abfragen. Mit Ausnahme der CLEAR- und der QUIT-Taste (ASCII-Code 2, bzw. 5), die zu einem sofortigen Programmabruch führen.

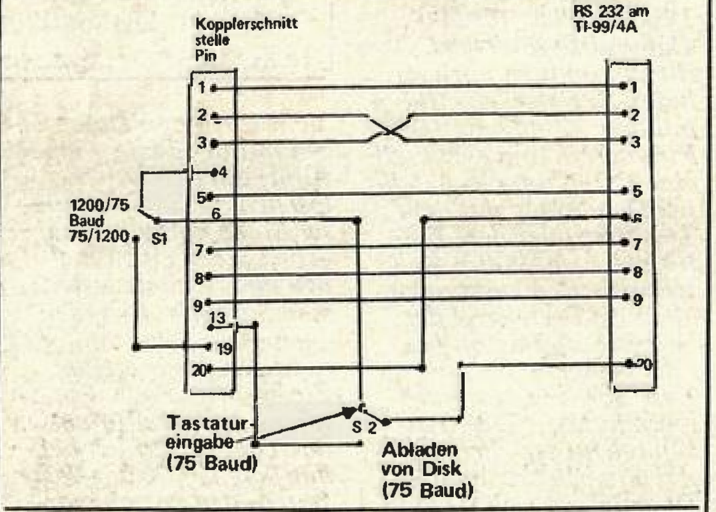
WELCHER DRUCKER FÜR DEN TI?

Da ich mir einen neuen Drucker zulegen möchte, bitte ich Sie um Rat, welchen Typ ich mir kaufen sollte.
Der Drucker müßte z.B. die Bilder von TI-Artist, Joy Paint und Business Graph-99 ausdrucken können und verschiedene Schriftarten beherrschen. Bitte schreiben Sie mir die Drucker auf, die diese Bedingungen erfüllen und schreiben Sie auch Preis und Bezugsort dazu.
M. Best, Gedern

Aus wettbewerbsrechtlichen Gründen dürfen wir Ihnen keine bestimmte Marke empfehlen. Jeder Fachhändler berät Sie in dieser Beziehung gerne weiter. Auch würde die von Ihnen gewünschte Liste bei dem breiten Druckerangebot sicherlich mehrere Seiten füllen. Im Prinzip erfüllt jeder Matrixdrucker ab einer Preisklasse von etwa 600,– Mark aufwärts Ihre Vorstellungen.
In den Programmen TI-Artist und Business-Graph-99 sind jedoch bereits Treiber für einige Druckermodelle installiert. Schreiben Sie sich die Namen dieser Drucker auf und legen sie dem Händler vor, damit er Ihnen ausschließlich Modelle vorführt, die zu diesen Typen kompatibel sind, d.h., über die gleichen Steuercodes angesprochen werden. Auch sollte sich der automatische Zeilenvorschub des Druckers über die sogenannten DIP-Schalter oder Jumpers wahlweise an- und ausschalten lassen. Dann dürften Sie mit dem Ausdruck Ihrer Grafiken keine Probleme haben.
Bedenken Sie dabei jedoch, daß Sie für den Anschluß am Centronics-Port Ihrer Schnittstellenkarte kein Standard-Druckerkabel verwenden können. Mit etwas Erfahrung im Umgang mit dem Lötkolben können Sie sich für wenig Geld selbst ein passendes Kabel basteln. Im anderen Fall nehmen Sie am besten die ganze Karte zum Händler mit

KORREKTUR „AKUSTIK-KOPPLER SONIC 300 CL

Leider sind beim Abdruck meines Beitrages in der TI-REVUE Nr. 1/87 kleine, jedoch für die Funktion entscheidende Fehler unterlaufen.



1. Im Text wird zwar darauf hingewiesen, daß die Pins 2 und 3 über Kreuz angeschlossen sind, auf der Zeichnung wurde die Kreuzung jedoch nicht durchgeführt.
2. Die Leitungsführung zum Schalter S 2 haben Sie etwas abgeändert, daraus ergibt sich aber eine Umkehrung der beiden Funktionen „Abladen von Disk" und „Tastatureingabe"
3. Der Anschluß-Lötpunkt an Pin 6 der Kopplerschnittstelle wurde vergessen.

VORZEITIGES ABSCHALTEN DER P-BOX

Zur Benutzung des TI-99 habe ich seit längerer Zeit zwei Fragen:
1.) Ich gebe ein Programm in die Konsole ein und schalte später die P-Box zu, um das Programm abzuspeichern. Dabei stürzt die Konsole bekanntlich ab. Gibt es einen Befehl, der das verhindert?
2.) Die Situation ist anders herum: Ich will die P-Box nur benutzen, um von Diskette ein Programm zu laden. Anschließend will ich diese abschalten. Dies verträgt der Computer genauso wenig. Gibt es eine Möglichkeit, die P-Box ordnungsgemäß abzumelden?
Reinhard Manthei, Kiel

In TI-Basic dürften Sie damit keinerlei Probleme bekommen. In dieser Grundsprache für den TI99 ist es durchaus möglich, ein Programm von Diskette zu laden und die Box danach auszuschalten. Der geräuschstarke Entlüfter wird damit zum Schweigen verdammt.
In einer anderen Programmiersprache ist dies nicht möglich, da ständig auf den Inhalt der Box (z.B. 32 KB-Erw.) zurückgegriffen werden muß. Auch in Extended Basic werden die Programme in die Erweiterung, soweit vorhanden, ausgelagert. Um Platz im Variablenspeicher zu schaffen, kann jedoch zumindest das Diskettenlaufwerk ausgeschaltet werden. Dies funktioniert sinngemäß über ein CALL FILES(0) und wird durch die Eingabe von CALL LOAD (–3188, 63,255) realisiert. Danach muß zumindest eine Programmzeile einmal editiert werden, um den vollen Stack-Speicher von 13928 Bytes zu erhalten. Dadurch haben Sie sich ganze 2 KByte hinzuverdient. Rufen Sie vor und nach dieser Operation die Funktion "SIZE" auf, können Sie den Hinzugewinn mitverfolgen. Das Diskettenlaufwerk gestattet nach dem Einschalten standardmäßig den gleichzeitigen Zugriff auf drei verschiedene Dateien. Dafür wird im Stack-Speicher Platz reserviert, so daß nur noch 11840 Bytes zur Verfügung stehen. Für größere Programme langt dieser Speicher natürlich nicht mehr aus. Durch Eingabe von CALL FILES(1) <ENTER> NEW <ENTER> kann gleichzeitig nur noch eine Datei offen gehalten werden, der freie Stack-Speicher steigt jedoch sprunghaft auf 12876 Bytes an.
Nun können auch Programme gefahren werden, die sich vorher auf Grund des zu geringen Speicherbereiches aufhängten oder erst überhaupt nicht laden ließen.
Wann Sie zu dieser Notlösung greifen müssen, zeigt Ihnen schon das Inhaltsverzeichnis der Diskette: Wird hier als Aufzeichnungsart nicht mehr "Program", sondern "Int/Var" angegeben, sollten Sie vor dem Laden erst zur Notbremse greifen.

XB-EXTENDER

Das größte Problem, dem ein X-Basic-Programmierer gegenübersteht, ist der Speicherplatz. Dieser ist, trotz inzwischen vieler erhältlicher Erweiterungen, immer noch auf ca. 32K beschränkt. Die Folge davon ist, daß viele Programme schon zum Scheitern verurteilt sind, ehe sie geschrieben wurden oder nur in einem sehr beschränkten Rahmen realisiert werden können. Ein Programm, das diesen Mißstand beseitigen soll, kam nun endlich auf den Markt: XB-Extender. XB-Extender macht es möglich, den zusätzlich mit der GRAM-Karte gewonnenen Speicherplatz zu nutzen. Der Mangel daran ist die große Anzahl von Peripheriegeräten, die das Hilfsprogramm voraussetzt. Es sind mindestens 128 GRAM, 32K Erweiterung, Extended Basic und mindestens ein Diskettenlaufwerk. Hat man jedoch alle Geräte zur Verfügung, so ermöglicht Extender, diese Geräte auch aus dem X-Basic heraus voll zu nutzen.
Ein Programm, das mit Hilfe von Extender erstellt worden ist, arbeitet, im Groben, folgendermaßen: Alle Programmteile, die das Hauptprogramm benötigt, sind im GRAM abgelegt. Diese können dann, unabhängig in der Reihenfolge, aufgerufen und gestartet werden. Der Vorteil hierbei: alle Variablenwerte bleiben erhalten. Doch dazu später mehr.
Zuerst einmal zum äußeren Aufbau und der generellen Funktionsweise des Programms. Extender ist in zwei Teilprogramme untergliedert. Ein Teil dient zum Erstellen der Teilprogramme im GRAM, der andere, um diese zu laden und zu starten. Diese Unterteilung bietet einen gravierenden Vorteil. Sie hilft Speicher sparen. Ist ein Programm im Speicher und im GRAM fertig erstellt, so muß man nur noch das Teilprogramm einladen, das zum Starten derselben dient. Der zweite Teil von Extender ist somit nicht mehr notwendig und belastet damit auch den Speicher nicht sinnlos.
Hat man nun mit Hilfe des ersten Teils ein Programm vollständig im GRAM untergebracht, so muß man diese Prozedur nicht vor jedem Programmstart wiederholen. Man kann nun die GRAM-Bereiche anhand der GRAM-Karten Loaders einfach auf Diskette speichern. Sie können dann zusammen mit dem Modul direkt wieder eingeladen werden. Der Start erfolgt dann durch ein Startprogramm, das man in X-Basic einfach schreiben kann. Doch nun genauer zu den einzelnen Programmteilen.

ERSTELLEN EINES PROGRAMMS IM GRAM

Um ein Programm so zu präparieren, damit es fehlerfrei aus dem GRAM heraus geladen und gestartet werden kann, benötigt man, wie schon erwähnt, den ersten Teil von Extender.
Damit dieser jedoch richtig funktioniert, muß man, wie wenn man das 'pre Scan' abschalten will, zuerst alle Variablennamen und CALL Unterprogramme im Hauptprogramm angeben. Also in dem Basic-Programmteil, der alle weiteren Teile aufruft und somit zuerst gestartet wird. Ist dies geschehen, so wird man, sobald das Programm läuft, für dieses Mehr an Arbeit schnell entlohnt.

DIE BEFEHLE VON GMOVE

GMOVE, das ist der Programmteil, mit dem man ein Basic-Programm erteilt, enthält mehrere Befehle für den Umgang mit dem neugewonnenen Spei-

TITEL AUS DEM VERLAG

ch erplatz. Dieser istin verschiedene Bereiche, den sogenannten 'Banks', geordnet. Bei der 128K-GRAM-Karte steht nur eine dieser Banks zur Verfügung. Diese lassen sich aber, je nach angeschlossenen GRAM-Karten, auf eine Anzahl von 14 Banks erweitern. Bei einer 128K-Karte stehen 32K zusätzlich zur Verfügung, was dem Speicherplatz dieser einen Bank gleichkommt. Ab 512K GRAM kann man sogar davon ausgehen, daß der Speicher nicht so schnell ausgeht, da die Banks nun 40K Speicher enthalten.
Die Bank der 128K-GRAM-Karte „faßt" nur 32K, da 8K für das ROM des X-Basic reserviert sind und somit für die eigene Anwendung nicht zur Verfügung stehen.
Bei 512K GRAM ist diese Bank ebenfalls nur mit 32K verwendbar. Alle anderen nutzen die vollen 40K GRAM aus.
Die Programme werden, und das ist ein weiterer Vorteil, nicht mit einer unendlich langen Adresse abgespeichert, sondern mit einem Namen und der Angabe der Bank, in die sie gelegt werden sollen. Pro Bank sind 16 Namen zugelassen, die, ähnlich wie auf Diskette, maximal 10 Zeichen lang sein dürfen.
An dieser Stelle zeigte sich der einzige Fehler, der bei Extender zu entdecken war. Hat man bei maximaler GRAM-Erweiterung alle Banks besetzt, so kommt man auf die stattliche Zahl von 224 Programmnamen. Verliert man nun seinen Aufschrieb, der alle Programme mit zugehöriger Bank katalogisierte, so darf man sich auf eine verzweifelte Suche begeben. Weder in GMOVE noch in GLOAD, das ist der Teil, der ein Programm aus dem GRAM lädt, ist ein Befehl integriert, der die Programmnamen, die in einer Bank enthalten sind, auflistet Es ist daher nützlich, sich alles aufs genau ste zu notieren.
Um nun endlich den zusätzlichen Speicherplatz nutzen zu können, muß man ihn zuerst, wie eine Diskette, initialisieren. Diese Initialisierung bewirkt, daß die angegebene GRAM-Bank aus dem „Module Library" herausgenommen wird, und so nicht zufällig durch ein Modul überschrieben werden kann, das man zusätzlich lädt. Sind alle Vorbereitungen abgeschlossen, so steht einem Ablegen der Basic-Teile nichts mehr im Wege. Diese können dann wie ein Modul abgespeichert und auch wieder geladen werden.

DAS STARTEN

Zum Starten des aufbereiteten Programms benötigt man den Programmteil 'GLOAD'. Dieser Programmteil enthält nur einen, dafür aber sehr umfangreichen, Befehl. Dieser erlaubt, neben der Angabe von Programmname und Bank, noch zwei weitere, sehr nützliche Parameter.
Einer dieser Parameter ist die Startzeilennummer. Also die Zeilennummer, bei der der neu geladene Programmteil starten soll. Die Angabe der Startzeilennummer ermöglicht es, in den neuen Programmteil zu springen, als ob dieser mit GOTO erreicht werden könnte.
Der zweite Parameter erlaubt die Angabe der DATA-Zeile, ab der ein Restore durchgeführt werden soll. Auch dieser Parameter erlaubt somit die Handhabung mit dem anderen Programmteil, als sei er ein „normaler" Teil des Hauptprogrammes.
Den krönenden Abschluß bildet die Ladezeit. Sie ist so kurz, daß sie nur als kleine Pause, die noch weit im Rahmen des Erträglichen liegt, wahrgenommen wird. Durch die Möglichkeiten und Arbeitsweise von Extender, ist kaum ein Unterschied festzustellen, ob es sich nun um ein einheitliches Programm oder um mehrere Programmteile handelt, die alle aus dem GRAM geladen werden.
Sieht man also vom Fehlen des Katalogs ab, so ist Extender ein rundum zufriedenstellendes Hilfsprogramm und somit fast ein Muß für jeden X-Basic-Programmierer, der eine GRAM-Karte zur Verfügung hat. Es ist beim
Elektronik Service
Linning 37
4044 Kaarst
zu erwerben.

Oliver Huber

TURBO PASCAL 99

Im Artikel bemängelte ich den unglücklichen Aufbau des Handbuches. Die Firma ‚Softpoint' teilte mir auf entsprechende Anfragen mit, daß dieses Handbuch hauptsächlich programmiererfahrene Anwender ansprechen soll. Anfang dieses Jahres soll ein Begleitbuch über Turbo Pascal-99 herauskommen, das auch Beispiele und erläuternde Kommentare enthalten soll. Der Preis liegt um DM 45,–, bei 250 Seiten Umfang.

Oliver Huber

TIPS & TRICKS

Den Bericht über den Sprachsynthesizer im Heft 3/4 '87 fand ich gut. Der Autor hätte sich aber nicht die Mühe machen müssen, TI in den USA anzuschreiben, um Datenblätter vom TMS 5220 zu erhalten.
Diese werden ebenfalls von TID herausgegeben. Außerdem liegt der Niederlassung in Freising ein in Deutsch verfaßter Applikationsbericht über die Sprachsynthese vor! In diesem Bericht wird u.a. der Aufbau der sog. „Lattice Filter" erklärt und detaillierte Informationen über die Befehle gegeben.
Ebenso hätte er die Befehle anstatt im Byte-Format im Bitmuster erklären sollen. Dies wäre für das Verständnis förderlich gewesen.
Ansonsten kann man nichts gegen den Bericht sagen, bis auf eins:
An einer Stelle wird geschrieben:
„Zusammen ergeben sie 50 Bits in einem Filter bei 40 Filter pro Sekunde." – Richtig müßte es so lauten: „Wenn alle Parameter für einen „FRAME" (das sind die 10 Filter, Energie, Tonhöhe und Wiederholungsbit) gesetzt wären, würde es eine Datenmenge von 40 Hz x 50 Bits = 2000 Baud betragen!"
Welche Bedeutung den einzelnen Filtern zukommt, möchte ich hier noch kurz ansprechen.
Durch das „LPC-Verfahren" (ein Sprachkodierverfahren) werden Worte digitalisiert, indem nach einem best. Zeitintervall Werte gemessen und dann gewandelt gespeichert werden.
Daraus resultiert kein absolut analoges Sprachsignal, sondern eine Art „zerhacktes" Signal (je nach dem wie groß oder klein das Zeitintervall ist.)
Dieses Signal (welches entweder im Sprach ROM fertig vorliegt oder erst selbst definiert werden muß) beeinflußt die zehn Filter. Jeder dieser Filter führt zwei Multiplikationen und zwei Additionen durch, die zur Interpolation der Zwischenwerte dient.
Ich bin sicher, daß nun endgültig alle Klarheiten beseitigt worden sind. Trotzdem nichts für ungut – im nachhinein ist man immer schlauer...

Oliver Siffrin

EINFÜHRUNG IN GPL-TEIL 4

Heute ist der Wurm drin. Wortwörtlich, sowohl im Programm selbst als auch in der Art, wie es mit dem TI umspringt. Denn heute geht es nicht sturheil weiter mit neuen Befehlen und neuen Operanden, obwohl das natürlich auch nicht zu kurz kommen darf. Ich zeige Ihnen zusätzlich anhand des Programms auch gleich ein paar Eigenheiten des Systems, genauer der PowerUp-Routine des Texanischen Instruments und wie Sie diese nutzen können.

Hin und zurück: CALL und RTN:
CALL ist ein ganz normaler Sprungbefehl, der aber seine Rücksprungadresse auf den GPL-Substack legt (und damit auch den Pointer >8373 um 2 hochzählt). Die Routine, auf die er springen soll, muß mit RTN als letztem Befehl abgeschlossen sein. Dieser RTN nimmt die oberste Adresse vom Substack und springt dorthin zurück. Beide Anweisungen setzen das C-Bit zurück! Übrigens: Alle GPL-Routinen, die Sie mit GPLLNK aus dem Maschinencode heraus erreichen können, erreichen Sie aus GPL heraus mit eben diesem CALL. So erzeugt z.B. CALL GROM&>0034 den Piepton. Erinnern Sie sich noch, was ich Ihnen bei der Beschreibung der Cassettenverwaltung mittels I/O über die Erzeugung der Recordermeldungen geschrieben hatte? Mit den „anderen Wegen" war auch dieser CALL gemeint, denn CALL GROM&>003B zusammen mit den üblicherweise für Dateiverwaltung zu setzenden Pointern und PABs spricht den Recorder so an, wie Sie es vom BASIC her gewohnt sind (siehe auch Handbuch zum E/A oder zur Mini-Memory).

CASE, beinahe wie ON X GOTO
CASE hat einen allg. Adreßoperanden und macht nichts weiter als das Byte, auf das der Operand zeigt, zweimal zur aktuellen GPL-Adresse zu addieren. Ist es z.B. Null, so passiert rein gar nichts. Ist es Eins, so überspringt der Interpreter die beiden Bytes nach CASE, bei Zwei überspringt er vier Bytes usw. Der Sinn davon: Nach CASE steht normalerweise eine Liste aus lauter BR-Statements (CASE setzt das C-Bit zurück!), die auf verschiedene Routinen zeigen. Und BR besteht nun einmal immer aus 2 Bytes. CASE und die anschließende Liste der BR-Befehle bilden somit eine Konstruktion, die im BASIC "ON X GOTO" heißt. Ein CASE existiert auch und sein Operand zeigt natürlich auf ein Wort.
Arithemetische und logische Befehle:
Bisher kennen Sie nur die Befehle aus dieser Gruppe, die nur einen Operanden haben. Deshalb hier der Rest:

GPL Befehl	Code (Bin/Hex)	GPL-Statusbyte	Maschinenbefehl
ADD	101000..(A0–A3)	CPU=> GPL	A, AB, AI
SUB	101001..(A4–A7)	CPU=> GPL	S, SB
MUL	101010..(A8–AB)	unbeeinflußt	MPY
DIV	101011..(AC–AF)	unbeeinflußt	DIV
AND	101100..(B0–B3)	CPU=> GPL	ANDI
OR	101101..(B4–B7)	CPU=> GPL	ORI, SOC(B)
XOR	101110..(B8–BB)	CPU=> GPL	XOR
ST	101111..(BC–BF)	unbeeinflußt	MOV(B), LI
EX	110000..(C0–C3)	unbeeinflußt	–

Der letzte Punkt in den Binärcodes steht wieder für das Bit, das die Wort- und Byteverarbeitung unterscheidet, das Bit davor legt fest, ob es sich bei dem zweiten Operanden des Befehls um einen allgemeinen Adreßoperanden (Bit=0) oder um einen IMM-Operanden (Bit=1) handelt. Wichtig: Im Gegensatz zum Maschinencode zeigt in GPL der erste Operand auf die Ziel-, der zweite auf die Quellenadresse, nicht umgekehrt! Des weiteren ist zu beachten, daß erstens dem Maschinenbefehl MPY in GPL nicht MUL sondern DMUL entspricht (MUL arbeitet ja nur mit Bytes!), was für DIV/DDIV ebenso gilt, und zweitens die im Maschinencode nicht vorhandene Anweisung EX(change) einfach Quelle und Senke vertauscht.

Statusbits kopieren mit H, G, CARRY und OVF
Bisher konnten Sie mit BR und BS nur das C-Bit des GPL-Status (gleich dem EQ-Bit des CPU-Status) auswerten. Da Statements wie ADD, AND, INC, ... aber immer den gesamten CPU-Status in das GPL-Statusbyte kopieren, müßte man doch an die restlichen Bits auch irgendwie herankommen. Und dies können Sie auch. H(igh) z.B. bewirkt, daß das C-Bit gleich dem "Logical-Greater-Bit" wird. Wollen Sie dieses Bit auswerten, so setzen Sie H vor den BR bzw. BS. Die Wirkung ist, als ob BR bzw. BS wirklich das Logical-Greater-Bit abgefragt hätte. Dementsprechend funktionieren GT für das "Arithmetik-Greater-Bit", CARRY für das "Carry-Bit" und OVF für das "Overflow-Bit".

MOVE kopiert ganze Speicherblöcke
MOVE ist in mehrfacher Hinsicht etwas ganz Besonders. Er ist der einzige Befehl, der gleich drei Operanden hat. Er ist auch der einzige Befehl, der die VDP-Register und GROMs/GRAMs ansprechen kann. Und er arbeitet nicht nur mit einzelnen Bytes oder Worten, sondern gleich mit ganzen Blöcken, wenn es sein muß. Er ist so vielseitig, daß ich auch ihn nicht auf einmal vollständig darlegen kann und möchte. Ich glaube sowieso, daß Sie an dieser Folge über GPL gegenüber allen anderen am meisten zu beißen haben, bis Sie sie vollständig durchschaut haben. Warum also unnötige Schwierigkeiten?
Die Aufgabe von MOVE wurde bereits erwähnt: Eine beliebige Anzahl von Bytes von einem Bereich in einen anderen zu kopieren. Bemerkenswert dabei schon die in ➔

Frage kommenden Speicherbereiche. Als Quelle kann dienen: Der ROM/RAM-Bereich, das VDP-RAM und GROMs/GRAMs. Als Senke sind zusätzlich noch die VDP-Register möglich. Die drei Operanden geben nacheinander folgendes an: Anzahl der Bytes, Startadresse der Senke, Startadresse der Quelle.
Die Form ist also:
MOVE Anzahl TO Senke FROM Quelle
Code: 001. ....

Die mittels Punkte dargestellten Bits bedeuten von links nach rechts:

a) Bit=0: Senke ist GRAM-Bereich.
Bit=1: Senke ist entweder VDP-Register oder durch einen allg. Adreßoperanden dargestellt (wird durch das folgende Bit entschieden).

b) Die Bedeutung dieses Bits hängt vom Wert des vorigen ab. Ist es Null, so ist es für heute für uns noch uninteressant und sollte auf Null gesetzt werden. Ist das vorige Bit Eins, so gilt für dieses:
Bit=0: Senke wird durch einen allg. Adreßo p. dargestellt (für RAM/ROM oder VDP-RAM).
Bit=1: Senke ist ein VDP-Register.

c) Bit=0: Quelle ist GROM/GRAM-Bereich.
Bit=1: Quelle wird durch einen allg. Adreßop. dargestellt (für RAM/ROM oder VDP-RAM).

d) Dieses Bit ist für uns heute noch unwichtig und bleibt deshalb auf Null.

e) Bit=0: Die Angabe der Anzahl der zu kopierenden Bytes liegt in einer Speicherstelle, die mit einem allg. Adreßop. adressiert wird. Der Wert für die Anzahl ist immer 16 Bits lang!
Bit=1: Die Anzahl gibt ein IMM-Operand (von 16 Bit Länge) an.

Das direkte Adressierungsverfahren erweitert: Die indizierte Adressierung

Das Prinzip ist schnell erklärt. Es handelt sich um einen ganz normalen direkten Operanden, der direkt auf irgendeine Adresse zeigt und der einfach mit einer zusätzlichen Angabe erweitert wurde. Bei diesem Zusatz handelt es sich um ein Byte, das mit >83 als Hi-Byte eine Adresse im Scratch-Pad-Ram (Bereich >8300 bis >83FF) ergibt. Und der Interpreter macht nichts weiter, als daß er zu der fest angegebenen Adresse des direkten Operanden (nicht deren Inhalt!) erst noch den Inhalt dieser Scratch-Pad-Adresse (Wort!) addiert, bevor er dann mit dem Ergebnis als neue Adresse seine Speicheroperationen durchführt.

VIII. DER KLEINE BEREICH – INDIZIERT

Der direkte Operand weist direkt auf eine Adresse im Bereich von >8300 bis >91FF (>8300-Offset nicht vergessen!). Das zusätzliche Byte zeigt (mit> 83 als Hi-Byte) auf ein Wort im Bereich> 8300 bis >83FF. Die Adresse des direkten Operanden plus dem Inhalt des Wortes ergibt nun eine neue Adresse, die auf jedes Byte von >0000 bis >FFFF zeigen kann.

Beispiel:
Sie haben den Operanden &>8323 (&>8357). Er besagt, daß der Interpreterzu >8323 das Wort aus >8357 addiert und dasErgebnis die fertige Adressebildet. Sie beschreiben ihn mit >C023 >57

Format:
1100 ++++ ....
direkter Op. *Zusatz*

IX. DER GESAMTE BEREICH – INDIZIERT

Der direkte Operand weist direkt auf jedes beliebige Byte im RAM/ROM (mit Offset!). An dem zusätzlichem Byte ändert sich nichts.

Beispiel:
Den Operanden &>1742(&> 8308) beschreiben Sie mit > CF >9442 >08.

Format:
11 0 01111.... ....
++++ *direkter Operand * *Zusatz*

Setzen Sie in den Formaten VIII oder IX das 3. Bit auf 1, dann gilt das dort gesagte selbstverständlich wieder für das VDP-RAM und der Offset entfällt.

Beispiel 1:
Sie geben für VDP&>0174(&>8317) die Hexcodes >E174 >17 ein.

Format X/1:
1110 + + ++.... ....
direkter Op. *Zusatz*

Beispiel 2:
Sie geben für VDP&>2906(&>83E2) die Hexcodes >EF >2906 >E2 ein.

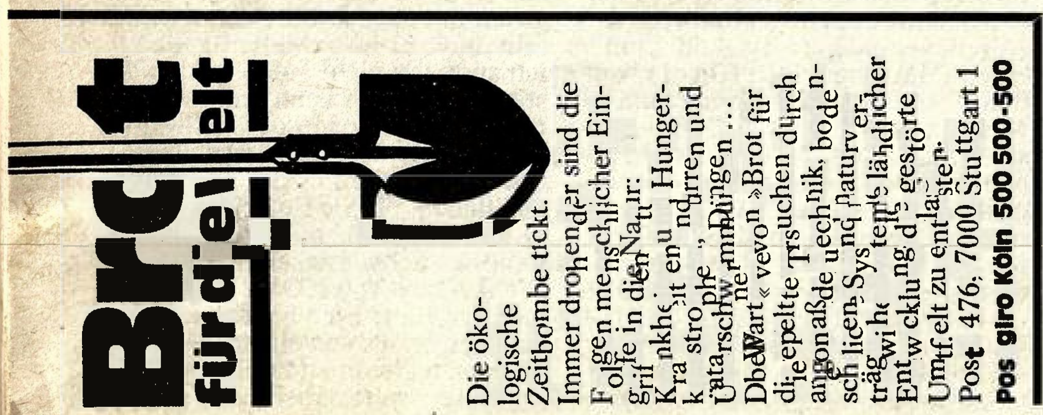
Format X/2:
1110 1111 ....
++++ *direkter Operand* *Zusatz*

DASPROGRAMM

Eigentlich sieht er ja mehr aus wie ein Mückenschwarm, unser Wurm. Worum es geht? Das Folgende ist nicht nur einfach wieder die übliche Demonstrationsspielerei, sondern würgt auch feste im System herum. Wir nutzen diesmal nämlich zuerst die Möglichkeit des Autostarts, verändern dann das Titelbild und bringen etwas Bewegung hinein, verlassen anschließend irgendwann das Programm unter Zuhilfenahme einer bisher noch ziemlich unbekannten Speicherstelle und nehmen nicht einmal EXIT dazu. Ein Wurm, der auf dem Titelbild herumspukt, das macht unser heutiges Programm. Beginnen wir mit dem Autostart...

```
G6000 AA80  0000 0000 0000
G6008 0000  0000 0000 0000
```

...und verändern danndas Titelbild, indem wir eine 23-Byte lange Textzeile aus dem GROMinden Bildschirmkopieren. Der GROM-Operand, den MOVE dazu benötigt, istnichtsweiter alsdie 16-Bit-GROM-Adresse ohne Veränderungen:



```
G6010 3100 MOVE >0017 TO
      17A1      FROM
      6560      GROM
      7C        &>607C
```

⪢6010

```
G6010 3100 MOVE>0017 TO
      17A1 VDP&>0165
      6560 FROM GROM
      7C   &>607C
```

Wie in der 1. Folge dieser Serie bereits ganz zu Anfang geschildert, geschieht der Autostart eines GPL-Programms noch bevor der Bildschirm sichtbar und die VDP-Interrupts zugelassen sind. Dies müssen wir hier deshalb selber tun und das VDP-Register 1 ist zuständig. Der Operand für ein VDP-Register ist nur ein einzelnes Byte mit der Registernummer:

```
G6017 3900 MOVE>0001 TO
      0101 REG>01 FROM
      6093 GROM&>6093
```

Bevor wir uns gleich den eigentlichen Routinen widmen, müssen wir unseren Wurm erst einmal auf das Titelbild zeichnen, indem wir 7 aufeinanderfolgende Positionen mit dem ASCII-Wert für das Sternchen überschreiben. Das, was vorher auf diesen Positionen stand, legen wir in eine Tabelle, um es nicht zu verlieren. Diese Tabelle enthält auch gleich die genaue Position von jedem der 7 Wurmglieder. Die Tabelle legen wir von >8304 bis >8318, die 4 Bytes davor benötigen wir für die Schleife, die diese Tabelle setzt.

```
G601DBF00 DST &>8300,
      0015 >0015
           Endpos. Tabelle wird
           rückw. gesetzt)
G6021 BF02 DST &>8302,
      0090 >0090
           Startpos. Bildschirm
G6025 BDC0 DST &>8401
      0100 (&>8300), &>8302
      02   Pos. e. Gliedes in
           Tabelle
G602A BCC0 ST &>8303
      0300 (&>8300),
      B002 VDP*>8302
           Alter Wert vom Bild-
           schirm in Tabelle
G6030 BEB0 ST VDP*>8302
      022A>2A
           Neuer Wert auf den
           Bildschirm
G6034 9102 DINC &>8302
           Nächste Bildpos.
G6036 A700 DSUB &>8300,
      0003 >0003
           Nächste Tab.pos.
G603A 4025 BR GROM&>6025
           Tab. vollständig?
           Nein: weiter
```

Für die Tastaturabfrage im Programm müssen wir >8374 (Tastaturmodus) noch auf >00 setzen, da uns das Betriebssystem beim Autostart in diesem Byte leider nur den unbrauchbaren Wert >01 hinterläßt:

```
G603C 8674 CLR &>8374
```

Die folgende Routine verhindert, daß von unserem Wurm Reste auf dem Bildschirm zurückbleiben, wenn er einmal über sich selbst gelaufen ist, was leider ständig passiert:

```
G603E BC00 ST &>8300,& 8306
      06
G6041 A600 SUB &>8300,>2A
      2A
G6044 604A BS GROM&>604A
```

Ist der Wurm nicht über sich selbst gelaufen, so wird sein letztes Glied vom Bildschirm gelöscht, indem auf die entsprechende Position wieder der ursprüngliche Wert kommt...

```
G6046 BCB0 ST VDP*>8304,
      0406    &>8306
```

...und dann verschieben wir die gesamte Tabelle um drei Bytes nach vorne. Die drei Bytes für das letzte (eben gelöschte) Glied fallen dabei aus der Tabelle heraus und am anderen Ende entsteht Platz für ein neues:

```
G604A 3500 MOVE >0012 TO
      1204 &>8304 FROM
      07   &>8307
```

Jetzt lassen wir den Computer ein neues Glied setzen, wobei die Richtung, in die es zeigen soll, zufällig ist:

```
G604F 0203 RAND> 03
G6051 8A78 CASE &>8378
G6053 4068 BR GROM&>6068
           ->hoch
G6055 4061 BR GROM&>6061
           ->runter
G6057 405D BR GROM&>605D
           ->rechts
G6059 9316 DDEC &>8316
           links
G605B 406C BR GROM&>606C
           Wert zu klein?
G605D9116 DINE &>8316
           rechts
G605F 406C BR GROM&>606C
           Sprung: weiter
G6061 A716 DSUB &>8316,>0020
           runter
      0020
G6065 0A   GT
           Wert zu klein?
G6066 606C BS GROM&>606C
           Nein: weiter
G6068 A316 DADD &>8316,>0020
      0020 hoch
```

Von dieser neu errechneten Position kopieren wir nun den dort gerade vorhandenen Wert in die Tabelle

```
G606C BC18 ST & 8318, VDP
      B016 *>8316
```

...und setzen auf die Position das Sternchen für das neue Glied:

```
G6070 03   SCAN
G6075 403E BR GROM&>603E
```

Auf Tastendruck soll der Spuk enden. Aber aufpassen! EXIT nutzt hier nichts, über den Autostart wären wir sofort wieder in unserem Programm. Wenn man aber weiß, daß der Aufruf eines Programms per Autostart mit der Anweisung CALL geschieht (was beim normalen Start nicht zutrifft!), dann ist es ganz klar, daß es mit RTN wieder zurück geht. In der Tat kommen

BEI TASTENDRUCK STOPPT DER WURM SOFORT

Sie mit RTN wieder zurück ins Betriebssystem und dort in die Tastaturabfrage des Titelbildes. D.h., Sie drücken eine Taste, und der Wurm stoppt sofort. Sie drücken noch eine Taste und sind im Hauptmenü. Das wollen wir aber gar nicht. Wir wollen von unserem Programm aus direkt ins Hauptmenü ohne die zweite Abfrage. Was tun? Wir überlisten das System einfach mit folgendem Trick: SCAN benutzt neben den allseits bekannten Speicherstellen noch einige weitere. >83C8 ist eine davon. Im Tastaturmodus 0 liegt dort die Kopie des Bytes, das SCAN in >8375 schreibt (im Modus 1 wäre es >83C9, in Modus 2>83CA, ...). Ob SCAN nun das C-Bit setzt oder nicht, hängt davon ab, ob die neue Taste den gleichen Code hat, wie er in >83C8 vom letzten SCAN hinterlegt wurde. Und wir sind so frevelhaft gemein, diese Speicherstelle auf >FF zu setzen, was für den folgenden SCAN bedeutet, es wurde gar keine Taste gedrückt:

```
G6077 BE80 ST &>83C8,>FF
      C8FF
G607B 00   RTN
```

Schließlich legen wir noch die Textdaten und den Wert für das VDP-Register 1 im Speicher ab und sind fertig:

```
G607C 48 49 45 53 20 49 53 54
G6084 20 44 45 52 20 57 55 52
G608C 4D 2044 52 49 4E 21
G6093 60
```

Sven Dyroff

DATEI-VERWALTUNG

Ein Datenverwaltungs-Programm dient zum Anlegen, Speichern und Laden von beliebigen Daten.
Die Daten, die von einem Datenverwaltungs-Programm verwaltet werden, bestehen aus einer Anzahl von Datensätzen, die gemeinsam eine Datei bilden. Die Struktur eines Datensatzes wird im Normalfall fest im Datenverwaltungsprogramm einprogrammiert (z.B. die beliebten Adress-Verwaltungsprogramme). Da aber der Wunsch besteht, beliebige Daten zu verwalten, ist es sinnvoll, die Art und den Aufbau der Daten getrennt vom Datenverwaltungsprogramm (DVP) zu beschreiben.
Man braucht also ein eigenes Datenverwaltungsprogramm und getrennt davon eine Beschreibung der zu verwaltenden Daten. Zur Erstellung dieser Beschreibungsdaten verwendet man einen Generator, der in der Lage ist, den Aufbau der zu verwaltenden Daten zu beschreiben. Das Datenverwaltungsprogramm muß die so erstellte Beschreibung der Daten (z.B. von Kassette) einlesen.
Wird also z.B. ein Adressverwaltungsprogramm benötigt, so braucht man eine Möglichkeit, auf einfache Weise den Aufbau eines Adress-Datensatzes zu beschreiben. Zusätzlich braucht man die Möglichkeit, den so beschriebenen Datensatz in einer Datei zu verwalten.
Die benutzerfrundlichste Art, einen Datensatz zu beschreiben, ist die Erstellung des Datensatzes interaktiv am Bildschirm. Man hat damit den Vorteil, daß der Aufbau eines Datensatzes so gewählt werden kann, wie die Daten später auch vom Datenverwaltungsprogramm dargestellt werden sollen. Da die Datensatzdarstellung auf dem Bildschirm auch als Maske bezeichnet wird, wird das Programm für die Erstellung eines Datensatzes auch Maskengenerator (MG) genannt.
Eine Maske besteht aus festen und variablen Daten. Die festen Daten sind die Namen oder Bezeichner der variablen Daten.
Beispiel:
Vorname: Fritz
Größe: 50
In diesem Beispiel sind die festen Daten "Vorname" und "Größe". Diese bleiben bei allen Datensätzen einer Datei gleich und bezeichnen das Nachfolgende variable Feld. Als variable Daten tauchen im obigen Beispiel die Werte "Fritz" (könnte auch "Hans", "Otto" oder "Susi" sein) und die Zahl "50" (die in unterschiedlichen Datensätzen unterschiedliche Werte haben wird) auf.
Die festen Daten und die maximale Länge der variablen Daten können mit Hilfe des Maskengenerators (MG) direkt auf dem Bildschirm erstellt werden. Der Maskengenerator erstellt aus der vom Benutzer am Bildschirm entworfenen Maske eine Datensatzbeschreibung und speichert diese auf Band(CS1) als ein komprimiertes File.
Nach dem Laden des Programms Datenverwaltung muß zum Anlegen einer neuen Datei als erstes diese Datensatzbeschreibung vom Kassettenrecorder (CS1) eingelesen werden. Dieser Vorgang kann über das Bedienmenü angewählt werden. Eine Datei besteht aus der Datensatzbeschreibung (feste Daten) und aus einer beliebigen Anzahl (Speichergröße!) von Datensätzen (variable Daten) die alle die Struktur haben, wie sie die Datensatzbeschreibung festlegt. (Die Anzahl der Datensätze ist als DIM Anweisung im Dateiverwaltungsprogramm bei Bedarf zu verändern. Standardwert sind 100 Sätze.)
Nach dem Aufbau mindestens eines Datensatzes kann die Datei gespeichert werden. Wird die Datei auf Band (CS1) gespeichert, so wird automatisch die Datensatzbeschreibung mitgespeichert. Genauso wird beim Einlesen die Datensatzbeschreibung wieder automatisch mit eingelesen. Das heißt, daß eine Datei immer aus Datensatzbeschreibung + Datensätzen besteht. Dadurch ist es möglich, mit dem Datenverwaltungsprogramm die unterschiedlichsten Daten (mit den unterschiedlichst aufgebauten Datensätzen) zu bearbeiten. Mit Hilfe des Maskengenerators ist es sehr einfach, einen für den jeweiligen Anwendungsfall benötigten Datensatz zu entwerfen.

Liebe TI-User,
Ihr mußtet Euch im letzten Heft nicht nur mit einem verminderten Umfang zufrieden geben, sondern zu allem Überfluß boten wir Euch im Listingteil auch noch einen Maskengenerator an, der ohne das eigentliche Hauptprogramm und die Beschreibung eigentlich keine Daseinsberechtigung besitzt.
Hier könnt Ihr nun den fehlenden Teil abtippen und lesen, wie alles funktioniert.
Damit uns keiner nachsagen kann, wir wollten damit finanziell nur unserem Disketten-Service etwas gutes tun, indem für ein komplettes Programm gleich zwei Disketten erworben werden müssen, haben wir sowohl das Vorprogramm wie auch das Hauptprogramm ohne zusätzliche Mehrkosten auf beide Disketten überspielt. Ihr habt also auf jeden Fall ein ablauffähiges Programm, ohne dafür zwei Hefte oder Disketten bestellen zu müssen.
Nochmals unsere Bitte um Entschuldigung.

Im folgenden wird der Aufbau der Daten, wie sie auf Band abgelegt, werden beschrieben. Mit dieser Information ist es möglich, die erstellten Daten mit anderen Programmen auszuwerten (sortieren, drucken usw.). Zum Einlesen, Anzeigen usw. können die Unterprogramme aus dem Datenverwaltungsprogramm übernommen werden.

Aufbau:

Jede Datei, die mit dem Dateiverwaltungsprogramm bearbeitet wird, hat folgenden Aufbau: Als erstes steht die Beschreibung eines Datensatzes auf Band. Diese Beschreibung wird mit dem Maskengenerator erzeugt, und kann von der Dateiverwaltung mit einem eigenen Kommando eingelesen werden.
Wenn die Dateiverwaltung eine Datei auf Band speichert, so legt sie die Datensatzbeschreibung (feste Daten) als erstes ab. Anschließend wird das Feld mit den einzelnen Nettodaten (variablen Daten) aufgezeichnet.
Sowohl bei der Datensatzbeschreibung als auch bei den Nettodaten handelt es sich um zweidimensionale Arrays (Felder), so daß für das Abspeichern und das Lesen eine Standardprocedure verwendet werden kann. Die beiden Proceduren (Write/Read) komprimieren die Arrays, so daß möglichst wenig Platz am Band belegt wird. Wesentlich wichtiger ist aber die Zeitersparnis, die dadurch erreicht wird. Die Standardprocedure Write speichert vor jedem Feld (Array) zwei Werte vom Typ numeric auf Band. Der erste Wert gibt den x-Wert des Array (Feld) an. Der zweite

LISTINGS

```
10Ø !**********************
11Ø !*    UNIVERSELLE      *
12Ø !*  DATEIVERWALTUNG    *
13Ø !*                     *
14Ø !* Benoetigte Geraete *
15Ø !*  TI-99/4A Konsole   *
16Ø !*     EXT.BASIC       *
17Ø !*   KASS.REC. CS1     *
18Ø !*                     *
216 !**********************
22Ø CALL SCREEN(12)
23Ø TA=1 !! Filedescriptor f
uer Tape
24Ø PR=2 !! Filedescriptor f
uer Printer
25Ø DIM DSA$(2Ø,4):: ! Aufba
u eines Datensatzes
26Ø DSX=Ø ! Anzahl der Maske
neintraege
27Ø DSY=Ø ! Elemente eines E
intrags (4 bzw 5)
28Ø DIM DAT$(1ØØ,15)!maximal
e Anzahl der Datensaetze der
 Datei(Speicherabhaengig)
29Ø HDS=1ØØ !! Groesste Date
nsatznummer
3ØØ VF=Ø !! Anzahl der Varia
blenfelder in einem Datensat
z
31Ø EF=Ø ! Dateiendezeiger v
orbesetzen
32Ø DISPLAY AT(12,7)ERASE AL
L:"DATEIVERWALTUNG"
33Ø FOR I=1 TO 1ØØØ :: NEXT
I
34 Ø !----------------------
-------------
35Ø ! Bedienmenue zur Progra
mm steuerung
36Ø !-----------------------
-------------
37Ø DISPLAY AT(2,4)ERASE ALL
:"B e d i e n m e n u e"
38Ø DISPLAY AT(5,4):"(1) - D
ateiparameter"
39Ø DISPLAY AT(8,4):"(2) - D
atensatz eingeben"
4ØØ DISPLAY AT(11,4):"(3) -
Datensatz anzeigen"
41Ø DISPLAY AT(14,4):"(4) -
Datei abspeichern"
42Ø DISPLAY AT(17,4):"(5) -
Datei einlesen"
43Ø DISPLAY AT(2Ø,4):"(6) -
DS-Beschr.einl."
44Ø DISPLAY AT(23,4):"(7) -
Ende"
45Ø !
46Ø CALL INKEY(K)
47Ø K=K-48
48Ø IF K<1 OR K>7 THEN 46Ø
49Ø CALL CLEAR
5ØØ ON K GOSUB 55Ø,86Ø,98Ø,6
3Ø,75Ø,74Ø,113Ø
51Ø GO TO 34Ø
52Ø !-----------------------
--
53Ø ! Dateiparameter anzeige
n
54Ø !-----------------------
--
55Ø DISPLAY AT(3,4):"Dateina
me: ":"":"      ";DSA$(Ø,Ø)
56Ø DISPLAY AT(8,4):"Datensa
etze/Feld:";VF
57Ø DISPLAY AT(1Ø,4):"Datens
atzanzahl :";EF
58Ø DISPLAY AT(12,4):"Max. D
s-Anzahl  :";HDS
59Ø DISPLAY AT(24,1):"Weiter
 mit Enter!" :: ACCEPT H$ ::
 RETURN
6Ø Ø--!---------------------
---
61Ø ! Datei auf band speiche
rn !
62Ø !-----------------------
---
63Ø IF EF=Ø THEN DISPLAY AT(
12,4)ERASE ALL:"Keine Datei
vorhanden!" :: GOTO 59Ø
64Ø DISPLAY AT(12,4)ERASE AL
L:"Kasette fuer Aufzeichnung
"
65Ø DISPLAY AT(14,4):"der Da
tei einlegen!"
66Ø DSA$(Ø,3)="1"
67Ø OPEN #TA:"CS1",OUTPUT,IN
TERNAL,FIXED 192
68Ø CALL WRITE(TA,DSA$(,),DS
X,DSY)
69Ø CALL WRITE(TA,DAT$(,),EF
,VF)
7ØØ CLOSE #TA :: RETURN
71Ø !-----------------------
----
72Ø ! Datei bzw DS-Beschr. l
esen
73Ø !
----
74Ø DISPLAY AT(14,4)ERASE AL
L:"Maskengener. erstellte" :
: GOTO 76Ø
75Ø DISPLAY AT(14,4)ERASE AL
L:"Dateiverwaltung erzeugte"
76Ø DISPLAY AT(12,4):"Eine m
it dem Programm"
77Ø DISPLAY AT(16,4):"Kasset
te einlegen!"
78Ø OPEN #TA:"CS1",INPUT ,IN
```

```
TERNAL,FIXED 192
790 CALL READ(TA,DSA$(,),DSX
,DSY)
800 VF=VAL(DSA$(0,4)):: EF=0
810 IF DSA$(0,3)="1" THEN CA
LL READ(TA,DAT$(,),EF,VF)
820 CLOSE #TA :: RETURN
830 !--------------------
840 ! Datensaetze eingeben
850 !
860 IF EF=HDS THEN DISPLAY A
T(12,4)ERASE ALL:"Datei voll
!"
870 IF VF=0 THEN DISPLAY AT(
12,4)ERASE ALL:"Es wurde kei
ne Datensatzbe- schreibung e
ingelesen!" :: GOTO 590
880 CALL MAUF(DSA$(,))
890 EF=EF+1
900 CALL MAEK(DSA$(,),DAT$(,
),EF,1)
910 DISPLAY AT(24,1):"Weiter
e Eingeabe(J/N)?:"
920 ACCEPT AT(24,25)SIZE(1)V
ALIDATE("J","N"):H$
930 DISPLAY AT(24,1):""
940 IF H$="N" THEN RETURN EL
SE 890
950 !-----------------
960 ! Ausgabe eines DS
970 !-----------------
980 IF EF=0 THEN DISPLAY AT(
12,4)ERASE ALL:"Kein Datensa
tz vorhanden!" :: GOTO 590
990 DISPLAY AT(24,6)ERASE AL
L:"Datensatz Nr.?"
1000 ACCEPT AT(24,21)BEEP VA
LIDATE(DIGIT)SIZE(3):NR
1010 IF NR>EF OR NR=0 THEN R
ETURN
1020 CALL MAUF(DSA$(,))
1030 CALL MAUS(DSA$(,),DAT$(
,)NR)
1040 DISPLAY AT(24,1)BEEP:"E
nde/Korr/Weiter(E/K/Enter)"
1050 ACCEPT AT(24,28)SIZE(1)
VALIDATE("E","K"):H$
1060 IF H$="E" THEN RETURN
1070 IF H$="K" THEN CALL MAE
K(DSA$(,),DAT$(,),NR,-1)
1080 NR=NR+1
1090 IF NR>EF THEN RETURN EL
SE 1030
1100 !-----
1110 ! Ende
1120 !-----
1130 CALL CLEAR :: DISPLAY A
T(20,4):"DATEIVERWALTUNG END
E" :: END
1140 !-----------------
1150 ! UP FUER CALL KEY
1160 !-----------------
1170 SUB INKEY(VA)
1180 CALL KEY(0,VA,S)
1190 IF S=0 THEN 1180
1200 SUBEND
1210 !-----------------
1220 ! Maske ausfuellen !
1230 !-----------------
1240 SUB MAUS(MA$(,),DS$(,),
DSNR)
1250 DISPLAY AT(1,25)SIZE(4)
:DSNR ! DSNR am BS anzeigen
1260 FOR I=1 TO VAL(MA$(0,1)
)
1270 ACCLAE=VAL(MA$(I,4))
1280 IF ACCLAE>0 THEN DISPLA
Y AT(VAL(MA$(I,1)),VAL(MA$(I
,3)))SIZE(ACCLAE):DS$(DSNR,I
-1)
1290 NEXT I :: SUBEND
1300 !---------------
1310 ! Maske aufbauen
1320 !---------------
1330 SUB MAUF(MA$(,))
1340 DISPLAY AT(1,1)ERASE AL
L:MA$(0,0)
1350 H=VAL(MA$(0,1))
1360 IF H=0 THEN BREAK ! Mas
ke leer: Fehler
1370 CALL CHAR(128,"0038")
1380 ST=VAL(MA$(0,2))!mit od
er ohne mark
1390 FOR I=1 TO H
1400 Z=VAL(MA$(I,1))
1410 S=VAL(MA$(I,2))
1420 SN=VAL(MA$(I,3))
1430 DISPLAY AT(Z,S)SIZE(SN-
S):MA$(I,0)
1440 IF ST=2 THEN CALL HCHAR
(Z+1,SN+2,128,VAL(MA$(I,4)))
1450 NEXT I :: SUBEND
1460 !-------------------------
-------------
1470 ! in Maske eingeben bzw
 korrigieren
1480 !-------------------------
-------------
1490 ! mod= 1 eingabe
      mod= -1 korrektur
1500 SUB MAEK(MA$(,),DS$(,),
NR,MOD)
1510 DISPLAY AT(1,25)SIZE(4)
:NR ! DSNR am BS anzeigen
1520 H=VAL(MA$(0,1))
1530 IF H=0 THEN BREAK ! Mas
ke leer: Fehler
1540 IF MOD<0 THEN 1590
1550 FOR I=1 TO H
1560 LAE=VAL(MA$(I,4))
1570 IF LAE>0 THEN DISPLAY A
T(VAL(MA$(I,1)),VAL(MA$(I,3)
```

LISTINGS

```
))SIZE(LAE):""
1580 NEXT I
1590 !
1600 FOR I=1 TO H
1610 LAE=VAL(MA$(I,4))
1620 IF LAE>0 THEN CALL BSEI
(VAL(MA$(I,1)),VAL(MA$(I,3))
,LAE*MOD,DS$(NR,I-1))
1630 NEXT I :: SUBEND
1640 !--------------------
1650 ! Hilfskonstruktion
1660 !--------------------
1670 SUB BSEI(Z,S,L,H$)
1680 ACCEPT AT(Z,S)SIZE(L)BE
EP:H$
1690 SUBEND
1700!--------------------
1710 ! Datei komprim. absp.
1720 !--------------------
1730 SUB WRITE(FD,ARRAY$(,),
XMAX,YMAX)
1740 PRINT #FD:XMAX,YMAX ::
ANZAHL=0
1750 FOR X=0 TO XMAX :: FOR
Y=0 TO YMAX
1760 LAEN=LEN(ARRAY$(X,Y))+1
1770 ANZAHL=ANZAHL+LAEN
1780 IF ANZAHL<190 THEN 1800
1790 PRINT #FD:"@" :: ANZAHL
=LAEN
1800 PRINT #FD:ARRAY$(X,Y),
1810 NEXT Y :: NEXT X :: PRI
NT #FD:"@" :: SUBEND
1820 !--------------------
1830 !Komprimierte Dat.les.
1840 !--------------------
1850 SUB READ(FD,ARRAY$(,),X
MAX,YMAX)
1860 INPUT #FD:XMAX,YMAX
1870 FOR X=0 TO XMAX :: FOR
Y=0 TO YMAX
1880 INPUT #FD:ARRAY$(X,Y,
1890 IF ARRAY$(X,Y)="@" THEN
 INPUT #FD:ARRAY$(X,Y):: GOT
O 1880
1900 NEXT Y :: NEXT X :: INP
UT #FD:H$ :: SUBEND
```

Wert gibt den y-Wert des Array (Feld) an.
Beispiel:
DIM MA$(x,y)
Im folgenden Bild wird der Aufbau der Felder beschrieben.
Datensatzbeschreibung (Maskenabbild arra)
Modus: im ersten Byte Angabe über Darstellung der Nettodatenfelder in der Maske.
B1=1 Keine Markierung des Nettodatenfeldes
B1=2 Markierung der Nettodatenfelder in der Maske.
Wird Markierung gewählt, so wird automatisch nur jede zweite Zeile zum Maskenaufbau verwendet, da die Zwischenzeilen für die Markierung benötigt werden. Dadurch ist es möglich, die Nettodatenfelder (Eingabefelder) mit einem Unterstrich zu kennzeichnen und dadurch die Länge des Eingabefeldes am Bildschirm sichtbar zu machen.

LISTINGS

DIE MAGISCHE 5. ZEILE

Immer wieder erreicht uns dieser verzweifelte Hilferuf der TI-User oder Einsteiger. In mehreren Ausgaben der TI-REVUE haben wir für dieses spezielle Problem bereits Hilfestellung geleistet. Da diese Ausgaben anscheinend an einigen Lesern vorbeigegangen sind und sich die mageren Dokumentationen von Texas Instruments vornehm darüber ausschweigen, nun nochmals die Erklärung:
In Extended-Basic ist es prinzipiell möglich, mehr als fünf Zeilen unter der gleichen Zeilennummer zu schreiben. Dies kann zum einen der Übersichtlichkeit des Programmes dienen (oder diese zerstören) und zum anderen wird damit Speicherplatz eingespart, der beim TI doch etwas knapp bemessen ist.
Am Ende der fünften Zeile bleibt der Cursor bei der normalen Eingabe mit einem ärgerlichen Piepston stehen und weigert sich strikt, die restlichen Befehle aus dem Heft aufzunehmen, die merkwürdigerweise noch unter der gleichen Zeilennummer stehen. Hier hilft ein ganz simpler Trick weiter. Aus dem Extended-Basic ist bekannt, daß man im Editier-Modus auch Zeilen duplizieren kann. Diese Funktion wird etwas umgewandelt auch für unser kleines Problem benötigt. Am Ende der fünften Zeile wird diese ganz normal mit der ENTER-Taste abgeschickt. Danach holt sie die Tastenkombination FCTN REDO wieder ins Leben zurück und der Cursor blinkt auf der Zeilennummer. Mit FCTN D wird die Markierung an das Ende der magischen 5. Zeile geführt und – ganz normal der Rest aus dem Heft abgetippt. Siehe da, der Cursor nimmt nun ohne Murren die restlichen Statements an.
Eines sollte dabei jedoch beachtet werden: Ein offenes Anführungszeichen muß zuvor stets geschlossen werden. Dazu einfach in der letzten Spalte der 5. Zeile ein (Hilfs-)Anführungszeichen setzen, welches anschließend wieder gelöscht und an der richtigen Stelle plaziert werden kann. Ansonsten spielt der Computer nicht mit und ärgert Sie mit einer weiteren Fehlermeldung.

HELI-SKIING

Landen Sie mit dem Helicopter im Gebirgstal, schnallen Sie sich die Ski an und wedeln Sie über die Hänge zur Schneebar. Wer sich als erster dort einen Jägertee genehmigen darf, der hat gewonnen. Die eingebaute Softwareuhr zählt unerbittlich jede Sekunde.
Achtung beim Fliegen! Es sind auch noch andere Hubschrauber unterwegs. Das Tempo kann mit den Tasten ‚1, 2, 3' bestimmt werden. Die Landung glückt, wenn die Feet-Anzeige genau NULL signalisiert. Jeder Hubschrauberabsturz kostet 5 Minuten. Ein Sturz beim Skifahren kostet 2 Minuten. Das Ziel ist erreicht, wenn Sie bis vor die Tür der Schneebar kommen.
Start mit roter Taste!

LISTINGS

```
10Ø !***********************
12 !                      *
13 !    * Heli-Skiing *   *
15 !       v 2.Ø          *
2Ø ! Copyright by NORBERT *
25 !    KALTENBRUNNER     *
3Ø !   CASTELLEZG.27/16   *
35 !     A-1Ø2Ø WIEN      *
4Ø !                      *
45 ! Benoetigte Peripherie*
5Ø ! TI99/4A+EX-MODUL+JOYS*
55 !                      *
6Ø !***********************
65 !
1ØØ CALL CLEAR :: KA=2 :: DI
M SI(3Ø):: DIM ZI(3Ø)
11Ø PRINT "JOYSTICK 2" :: PR
INT
13Ø PRINT "ALPHALOCK AUSRAST
EN" :: PRINT :: PRINT "REGEL
N J/N ?"
136 CALL KEY(Ø,KKK,SSS):: CA
LL KEY(2,MMM,GGG):: IF MMM=1
8 THEN 14Ø
137 IF KKK=74 OR KKK=1Ø6 THE
N 41ØØ
138 IF KKK=78 OR KKK=11Ø THE
N 14Ø
139 GOTO 136
14Ø FELS$="1Ø1Ø183838383C3C7
D7D7DFFFFFFFFFFØØØØØ4Ø4Ø4Ø6Ø
6ØEØE8E8E9FDFDFFFFF"
145 PRINT :: PRINT "BITTE WA
RTEN"
15Ø HUET$="ØØØF152A55FF4Ø5E5
2525E4Ø4ØFF4949Ø4F45CAC56FFØ
2F29292D29292FF2424"
16Ø BARS$="ØØFFØØ714A4A734A4
A72ØØFF1Ø1Ø1Ø1ØØØFFØØ9E5252D
E58545200FF1Ø1Ø1Ø1Ø"
17Ø EXPL$="Ø8442212492512ØF1
163450912244408081122244851A
6C8F4C2615Ø48442210"
18Ø SKI1$="ØØØØØØ1ØØ8Ø5Ø3Ø76
633190FØ6Ø3Ø1ØØØØØØØØØ6Ø6Ø8Ø8
ØØØ8Ø4Ø8Ø8ØØØØØØ8ØEØ"
19Ø SKI2$="ØØØØØØØ6Ø6Ø1Ø1ØØØ
1Ø2Ø1Ø1ØØØØØØ1Ø7ØØØØØØØ81ØAØC
ØEØ66CC98FØ6ØCØ8ØØØ"
2ØØ SKI3$="274845484B6F35191
DØ643E346ØC183ØCØ44426256CC1
D3265CØ8Ø82C7623Ø18"
21Ø F$="Ø4ØC1C3C1CØCØ4Ø4Ø4Ø4
Ø4Ø4Ø4Ø4Ø4ØØØØØØØØØØØØØØØØØØ
ØØØØØØØØØØØØØØØØ"
22Ø BAUM$="Ø1Ø3Ø7Ø3Ø7ØFØ7ØF1
FØ7ØF1F3F39Ø1Ø1ØØ8ØCØ8ØCØEØC
ØEØFØCØEØFØF838ØØØØ"
23Ø BLANK$=" "
24Ø HUBLI$="ØØØØØØØØØØFDØ21F
23478E7FØ89Ø7FØØØØØØØØØØØØØF8
ØØC1E2FFC28180Ø4ØF8ØØ"
25Ø STU$="12Ø9ØFØØØØØ5Ø81Ø18
171Ø1Ø2428418424489ØFØ8Ø8ØCØ
2Ø6ØAØ2Ø1Ø9Ø1ØØ828"
26Ø HUBRE$="ØØØØØØØØØØØ1FØØ83
47FF4381Ø1Ø21FØØØØØØØØØØØØØBF
4ØF8C4E271FE1ØØ9FEØØ"
27Ø CALL CHAR(96,SKI2$,1ØØ,S
KI3$,1Ø4,SKI1$,116,F$,12Ø ,BA
UM$,13Ø,BLANK$,1Ø8,HUBLI$,11
2,HUBRE$,136,STU$)
275 H,M,SEC=Ø
28Ø CALL COLOR(9,9,1,1Ø,9,1,
11,9,1,12,13,1,13,16,16)
29Ø FOR C=2 TO 8 :: CALL COL
OR(C,2,12):: NEXT C
3ØØ CALL CHAR(92,FELS$,14Ø,H
UET$)
31Ø CALL COLOR(8,16,1)
32Ø CALL MAGNIFY(3):: RANDOM
IZE
33Ø CALL CLEAR :: CALL SCREE
N(2)
34Ø DISPLAY AT(1,1):"xzxzxzx
zxzxzxzxzxzxzxzxzxzxzy{y{y{y
{y{y{y{y{y{y{y{y{y{y{"
35Ø FOR Z=3 TO 21 STEP 2
36Ø W$=SKI1$ :: DISPLAY AT(Z
,1):"xzhjhjhjhjhjhjhjhjhjhjh
jhjxzy{ikikikikikikikikikiki
kiky{" :: NEXT Z :: DISPLAY
AT(23,1):"xzxzxzxzxzxzxzxzxz
xzxzxzxzxzy{y{y{y{y{y{y{y{y{
y{y{y{y{y{" :: GOSUB 136Ø
37Ø IF W$=SKI1$ THEN W$=SKI2
$ ELSE W$=SKI1$
38Ø IF S=4 THEN S=1 ELSE S=S
+1
39Ø IF COL=16 THEN COL=3 ELS
E COL=COL+1
4ØØ FOR Z=1 TO 4Ø :: NEXT Z
:: CALL CHAR(1Ø4,W$):: CALL
COLOR(#S,COL):: CALL SOUND(-
15ØØ,-5,5,111,5,5678,5):: CA
LL KEY(Ø,K,SSS)
415 IF K=49 THEN T=1 :: GOTO
 43Ø ELSE IF K=5Ø THEN T=1.5
 :: GOTO 43Ø ELSE IF K=51 TH
EN T=2 :: GOTO 43Ø ELSE 37Ø
42Ø GOTO 37Ø
43Ø CALL SOUND(5ØØ,44Ø,1)::
CALL SOUND(5ØØ,66Ø,1):: CALL
 SOUND(1ØØØ,88Ø,1):: HUB=6 :
: CALL CHAR(1Ø4,SKI1$)
44Ø CALL DELSPRITE(ALL):: CA
LL CLEAR :: CALL SCREEN(8)
45Ø KA=K-48 :: CALL VCHAR(3,
1,13Ø,1Ø):: CALL VCHAR(7,2,1
3Ø,6):: CALL VCHAR(1,1,92,1)
:: CALL VCHAR(2,1,93,1)
```

```
460 CALL VCHAR(5,2,94,1):: C
ALL VCHAR(6,2,95,1)
470 CALL VCHAR(5,32,130,12):
: CALL VCHAR(11,31,130,4)::
CALL VCHAR(3,32,92,1):: CALL
 VCHAR(4,32,93,1):: CALL VCH
AR(9,31,92,1)
480 CALL VCHAR(10,31,93,1)
490 CALL HCHAR(13,1,130,384)
500 DISPLAY AT(11,3):"\^\^\^
\^\^\^\^\^\^\^\^\^]_" :: D
ISPLAY AT(10,28):"\^"
510 LPO=INT(3*RND)+1
520 GOSUB 1070
530 IF LPO<>1 THEN 560
540 DISPLAY AT(11,1)SIZE(3):
"   " :: DISPLAY AT(12,1)SIZ
E(3):"   "
550 B1=19 :: B2=33
560 DISPLAY AT(22,1)SIZE(32)
:"*FEET*SPEED*HELIS*********
**" :: DISPLAY AT(24,1):"***
*************************"
570 VAR=108 :: VA2=112 :: CA
LL COLOR(11,5,12)
580 FOR Z=1 TO HUB-1 :: IF Z
=1 THEN SD=19 ELSE SD=SD+2
590 DISPLAY AT(23,SD)SIZE(2)
:"pr" :: DISPLAY AT(24,SD)SI
ZE(2):"qs" :: NEXT Z
600 CALL SOUND(300,1500,1)::
 CALL SPRITE(#1,VAR,2,10,230
)
610 CALL SPRITE(#3,112,14,50
,20,0,10+INT(5/HUB),#4,112,3
,30,3,0,7+INT(5/HUB),#5,112,
9,10,10,0,12+INT(5/HUB))
630 DISPLAY AT(23,13)SIZE(3)
:HUB :: DISPLAY AT(23,8)SIZE
(3):KA
650 GOTO 1020
710 ! # BILD 2 #
720 HC=120 :: E=0 :: DISPLAY
 AT(1,16):"CORRECT LANDING"
:: CALL MOTION(#1,0,0):: FOR
 SOU=HI TO 1 STEP -1 :: CALL
 SOUND(100,-5,3,110+SOU,3,46
55+(5*SOU),3):: NEXT SOU
730 FOR DELAY=2 TO 15 STEP -
1 :: CALL SOUND(200,-5,DELAY
):: NEXT DELAY :: CALL DELSP
RITE(ALL)
740 CALL CLEAR :: CALL SCREE
N(16):: T=4 :: VAR=96 :: VA2
=104 :: BILD=2
750 CALL SPRITE(#1,VA2,9,70,
100)
760 GOSUB 5000
780 CALL SPRITE(#11,HC,4,118
,100):: V=5 :: D=1
785 IF HC=120 THEN HC=136 EL
SE HC=120
790 CALL SPRITE(#3,120,3,111
,20)
800 CALL SPRITE(#4,136,2,118
,119)
820 CALL SOUND(100,1000,6)
825 DISPLAY AT(24,1):"     P
RESS RED BUTTON!"
830 CALL KEY(2,KK,SS):: GOSU
B 5000 :: IF SS=0 THEN 830
835 DISPLAY AT(24,1):""
840 CALL JOYST(2,X,Y):: CALL
 MOTION(#1,-Y*T,X*T):: CALL
PEEK(-31877,U):: IF U=188 TH
EN 1300 ELSE CALL PEEK(-3187
7,U):: IF U=188 THEN 1300
850 IF X<=0 THEN CALL PATTER
N(#1,VAR)ELSE CALL PATTERN(#
1,VA2)
860 CALL PEEK(-31877,U):: IF
 U=188 THEN 1300 :: CALL SOU
ND(675,-5,14):: CALL POSITIO
N(#1,XX,YY,#11,F8,F9):: IF F
8<173 THEN 880
870 CALL PEEK(-31877,U):: IF
 U=188 THEN 1300 :: CALL LOC
ATE(#11,219,YY)
880 GOSUB 5000 :: CALL PEEK(
-31877,U):: IF U=188 THEN 13
00 :: E=E+1 :: V=V+2*INT(D/1
0):: IF V>60 THEN V=35
890 CALL MOTION(#11,-V,0,#3,
-V,0,#4,-V,0):: CALL PEEK(-3
1877,U):: IF U=188 THEN
1300 :: D=D+1 :: IF E>100 TH
EN 3000 ELSE 840
960 ' STEUERUNG BILD 1
1020 D=D+1 :: GOSUB 5000 ::
CALL SOUND(-2000,-5,3,120+HI
,3,4655+HI,3):: CALL JOYST(2
,X,Y):: CALL MOTION(#1,-Y*T,
X*T)
1025 FOR C3=3 TO 5 :: CALL C
OINC(#C3,#1,15,U):: IF U=-1
THEN 1220
1027 NEXT C3
1030 CALL POSITION(#1,HI,B):
: DISPLAY AT(23,2)SIZE(3):82
-HI :: IF HI>=83 OR B>239 TH
EN 1220
1035 IF HI>67 THEN IF B<B1 O
R B>B2 THEN 1220 ELSE 1062
1040 IF X<=0 THEN CALL PATTE
RN(#1,VAR)ELSE CALL PATTERN(
#1,VA2)
1055 FOR C3=3 TO 5 :: CALL C
OINC(#C3,#1,15,U):: IF U=-1
THEN 1220
1057 NEXT C3 :: GOTO 1020
1062 CALL JOYST(2,X,Y):: CAL
```

LISTINGS

```
L MOTION(#1,-Y/7,X/7):: CALL
 POSITION(#1,HI,B):: DISPLAY
 AT(23,2)SIZE(3):82-HI :: IF
 HI=82 AND B>=B1 OR HI=82 AN
D B<=B2 THEN 720 ELSE IF HI>
82 THEN 1220 ELSE IF HI<68 T
HEN 1020
1069 CALL SOUND(-500,-5,3,14
0+HI,3,4655+HI,3):: GOSUB 50
00 :: GOTO 1062
1070 ' # BILD 1 #
1080 FOR SPA=1 TO 240 STEP 2
0
1090 DISPLAY AT(11,SPA)SIZE(
2):"xz" :: DISPLAY AT(12,SPA
)SIZE(2):"y{"
1100 NEXT SPA
1110 IF LPO=1 THEN 1140
1120 DISPLAY AT(11,3)SIZE(5)
:"    \" :: DISPLAY AT(12,3)
SIZE(5):"    ]"
1130 B1=32 :: B2=46
1140 'BAEUME
1150 FOR S=11 TO 21
1155 READ ZI(S),SI(S)
1170 IF S=20 THEN CALL SPRIT
E(#S,140,14,ZI(S),SI(S)):: G
OTO 1210
1180 IF S=14 THEN CALL SPRIT
E(#S,136,2,ZI(S),SI(S)):: GO
TO 1210
1190 CALL SPRITE(#S,120,13,Z
I(S),SI(S))
1210 NEXT S :: RESTORE :: RE
TURN
1220 F=1 :: CALL COLOR(#1,9)
1230 F=F+2 :: CALL MOTION(#1
,F,X):: COLI=0 :: CALL SOUND
(-100,-5,0,2000-F*10,2):: CA
LL POSITION(#1,HI,B):: IF HI
<86 THEN 1230 ELSE CALL MOTI
ON(#1,0,0)
1231 CALL SOUND(-999,-5,2)::
 CALL DELSPRITE(#3,#4,#5)::
CALL HCHAR(23,19,42,12):: CA
LL HCHAR(24,19,42,12)
1235 CALL CHAR(112,EXPL$,108
,EXPL$):: FOR SOU=0 TO 10 ST
EP 2 :: CALL SOUND(100,-6,SO
U):: NEXT SOU
1240 CALL SCREEN(11):: CALL
SOUND(-600,-6,COLI,1000-COLI
*30,COLI):: CALL SCREEN(11):
: COLI=COLI+1 :: CALL SCREEN
(10)
1250 IF COLI=7 THEN 1260 ELS
E 1240
1260 CALL DELSPRITE(#1):: HU
B=HUB-1 :: M=M+5 :: IF HUB=0
 THEN 1450
1270 FOR DELAY=1 TO 20 :: NE
XT DELAY
1280 CALL CHAR(108,HUBLI$,11
2,HUBRE$):: CALL COLOR(#1,2)
1290 CALL SCREEN(8):: GOTO 5
80
1300 CALL COINC(#1,#6,20,U):
: IF U=-1 THEN CALL MOTION(#
1,0,0):: GOTO 4000 ELSE CALL
 SCREEN(2) !KOLLROUTINE BILD
2
1310 CALL SOUND(1000,-6,0)
1320 CALL PATTERN(#1,100)::
CALL MOTION(#1,0,0):: CALL M
OTION(#11,0,0):: DISPLAY AT(
24,1):"COLLISION" :: CALL MO
TION(#3,0,0):: CALL MOTION(#
4,0,0):: M=M+2
1330 FOR DELAY=1 TO 15 :: GO
SUB 5000 :: CALL SOUND(-300,
2000/DELAY,DELAY):: NEXT DEL
AY
1340 CALL DELSPRITE(#11,#4,#
3,#1)
1350 GOTO 740
1360 ! TITELBILD
1370 S=1 :: COL=3
1380 CALL HCHAR(3,5,42,24)::
 CALL VCHAR(4,5,42,18):: CAL
L HCHAR(22,5,42,24):: CALL V
CHAR(4,28,42,18)
1390 DISPLAY AT(11,8)SIZE(13
):"*******************" ::
DISPLAY AT(12,8)SIZE(13):"*H
ELI-SKIING*"
1400 DISPLAY AT(13,8)SIZE(13
):"*************" :: DISPLAY
 AT(22,8)SIZE(12):"PRESS*BUT
TON"
1410 DISPLAY AT(9,8)SIZE(12)
:"^xzxzxzxz\^\^" :: DISPLAY
 AT(10,8)SIZE(12):"]_y{y{y{y
{]_]_"
1420 CALL SPRITE(#1,112,8,15
2,50,0,15,#2,112,14,110,60,0
,20,#3,112,16,24,243,0,25,#4
,112,11,43,190,0,10)
1430 DISPLAY AT(19,8)SIZE(13
):"*SPEED-1-2-3*" :: DISPLAY
 AT(17,19)SIZE(2):"xz" :: DI
SPLAY AT(18,19)SIZE(2):"y{"
1440 RETURN
1450 DISPLAY AT(23,6):"GAME
OVER"
1460 GOSUB 5000 :: DISPLAY A
T(4, 2):"* USED TIME FOR HELI
COPTER
                 ACCIDENTS *"
1480 CALL KEY(2,K,ST):: IF K
=18 THEN CALL DELSPRITE(ALL)
:: CALL CLEAR:: CALL CHARSE
```

```
T :: RUN 14Ø ELSE CALL KEY(Ø
,KKK,ST):: IF KKK=1Ø1 THEN E
ND ELSE 148Ø
3ØØØ T=3 ! ENDE
3Ø3Ø CALL JOYST(2,X,Y):: CAL
L MOTION(#1,-Y*T,X*T):: CALL
 PEEK(-31877,U):: IF U=188 T
HEN 13ØØ
3Ø35 CALL SOUND(-1ØØØ,-5,14)
3Ø4Ø IF X<=Ø THEN CALL PATTE
RN(#1,VAR)ELSE CALL PATTERN(
#1,VA2):: GOSUB 5ØØØ
3Ø5Ø CALL MOTION(#3,-V,Ø,#4,
-V,Ø):: IF V=3 THEN 3Ø55 ELS
E 3Ø6Ø
3Ø55 CALL CHAR(1Ø8,BARS$)::
CALL SPRITE(#5,1Ø8,3,14Ø,7Ø)
3Ø6Ø IF V<=Ø THEN CALL SPRIT
E(#7,12Ø,4,118,15Ø):: CALL S
PRITE(#6,14Ø,2,14Ø,15Ø):: CA
LL MOTION(#3,Ø,Ø,#4,Ø,Ø,#11,
Ø,Ø)ELSE V=V-1
31ØØ GOTO 3Ø3Ø
4ØØØ ! GESCHAFFT
4Ø1Ø GOSUB 5ØØØ :: DISPLAY A
T(22,2):"* GRATULATIONS, SNO
WBAR IS     REACHED, CHEERS!
*"
4Ø15 FOR SOU=11Ø TO 11ØØ STE
P 1ØØ :: CALL SOUND(2ØØ,SOU,
3,SOU+1ØØØ,3):: NEXT SOU ::
CALL SOUND(8ØØ,121Ø,3,221Ø,3
)
4Ø17 DISPLAY AT(12,2):"PRESS
 BUTTON FOR NEW GAME"
4Ø2Ø CALL KEY(2,KK,ST)
4Ø3Ø IF KK=18 THEN CALL DELS
PRITE(ALL):: CALL CLEAR :: C
ALL CHARSET :: RUN 14Ø
4Ø5Ø GOTO 4Ø2Ø
41ØØ ! REGELN
41Ø5 CALL CLEAR :: CALL SCRE
EN(4):: PRINT "     * HELI-S
KIING *" :: PRINT :: PRINT :
: PRINT
411Ø PRINT "LANDEN SIE MIT D
EM HELI-    COPTER IM GEBIRG
STAL. WAEH- LEN SIE DIE PASS
ENDE GE-    SCHWINDIGKEIT MI
T DEN TAS-"
412Ø PRINT " TEN '1,2,3' . D
IE ANZEIGE  'FEET' GIBT DIE
HOEHE AN.   IST 'FEET' GENAU
 NULL,DANN  KOENNEN SIE MIT
DER SKI-"
413Ø PRINT "ABFAHRT ZUR SCHN
EEBAR BE-   GINNEN. WER DAFU
ER DIE      KUERZESTE ZEIT B
ENOETIGT,   DER HAT GEWONNEN
."
414Ø PRINT :: PRINT "JEDER H
UBSCHRAUBERABSTURZ  BRINGT
5, JEDER SKIFAHRER- STURZ E
RGIBT 2 STRAFMINUTEN STEUERU
NG:JOYST.2" :: PRINT
415Ø PRINT "  * ROTE TASTE D
RUECKEN *"
416Ø CALL KEY(2,KKK,SSS)
417Ø IF KKK=18 THEN CALL CLE
AR :: GOTO 14Ø
418Ø GOTO 416Ø
5ØØØ ! UHR
5Ø1Ø CALL PEEK(-31879,ZN)::
Z=Z+ZN-ZA-255*(ZA>ZN):: IF Z
<3ØØØ THEN 5Ø3Ø
5Ø2Ø Z=Z-3ØØØ :: M=M+1 :: IF
 M=6Ø THEN H=H+1 :: M=Ø
5Ø3Ø SEC=INT(Z/5Ø):: ZA=ZN :
: DISPLAY AT(1,1):H;":";M;":
";SEC
5Ø4Ø RETURN
6ØØØ ! BAUMPOSITIONEN
6Ø1Ø DATA 1Ø6,188,123,153,1Ø
5,32,127,85,11Ø,135,148,144,
136,224,149,131,12Ø,2Ø6,122,
59,146,15
```

LISTINGS

BÖRSE

TI REVUE

Verkaufe: Adapter für Trackball, Platine fertig geätzt, gebohrt, bestückt (s. Revue 1/87) f. 34,90 DM incl. Porto u. Verp. Frank Müller, 5014 Kerpen-Türnich, Ursfelderstr. 49, Tel. 02237/8278

Verk.: 1 TI99/4A, P-Box, RS232, P, 36K, 1 LW, Ex.B., EA, Apes. GR.-Prog., DFÜ TE 1200, 1 Modul, ca. 60 Prog. v. K. Hagenbuchner, B.-Kurs, jede Menge Lit., Epson RX80, alles orig. verp., VB 2.350 DM. T. 02238/50517 ab 18 Uhr

---

Verkaufe wg. Systemwechsel TI99/4A, CPS 99, Joysticks, Centronics-Kabel, Othellomodul, Software, Lit., Ext. Basic, VB 1.200 DM. Tel. 0201/680403

Verkaufe 99 Spezial I u. II, TI99/4A Intern, TMS9900 f. Mini Mem, TI Writer, Komp., Disk. Rechnungsstellung, TI Special Sonderheft 2/85, Module Pers. Rec. Keeping u. Report Generator, Preis VB., Tel. 06764/2425

32K CMOS-RAM-Erw., Akkupff., seitl. ansteckb., Bus durchgef. 110,—; Quicksave-Modul, seitl. ansteckb., 6 x schneller auf Kass. speichern, neue Basic-Bef. 80,—. S. Schulz, Tel. 0621/474692

Suche für den TI99/4A Prog. f. Heizung, Lüftung, Klima- u. Sanitärtechnik, z.B.: Wärmebedarfsber. nach DIN 4701, K-Zahlber. m. Temp.-Verkauf, Kanalnetzber. nach DIN. Angebote an F. Bennert, T. 0731/382159 ab 19 Uhr

---

Verkaufe: 32KBytes ext. m. Bus! 110,—; Munchman (15) M.A.S.H 30,—; Buch: Superspiele f. TI 10,— u. Vieles mehr! Liste anfordern! Sascha Will, Lindenstr. 15, 7162 Gschwend, 07972/431

*** Achtung *** Verkaufe * seikosha GP 100A, MBI Interface f. 250 DM, 4A-Laufwerk f. 200 DM. David Hesshaimer, Tel. 08806/7372. Anrufe werden von 19-21 h entgegengenommen.

FRONT

Ob sich der Spieler für die Bundeswehr eignet und bis zu welchem Rang er aufsteigen kann, will dieses Geschicklichkeitsspiel in verschiedenen Schwierigkeitsstufen herausfinden.
Aufgabe des Spielers ist, den Nachschub eines fiktiven Feindes zu unterbrechen. Auf alle erdenklichen Arten versucht der Gegner, seine Munitions- und Waffentransporte in die Kriegszone zu befördern: Mit Flugzeugen, auf dem Schienenweg, zu Wasser, mit Lkw's und Panzern.
Mit Hilfe des Joystick in Port 1 und eines Fadenkreuzes sollen die Transporte nun ins Visier genommen werden. Keine einfache Sache, denn der Gegner hält natürlich nicht still. Reaktion und Treffsicherheit sind die entscheidenden Faktoren, um mit einem einigermaßen guten Prozentsatz abzuschneiden. Die Anzahl der Schüsse ist nicht vorgeschrieben. Der Spieler wählt beliebig zwischen 1–99 Schuß aus. Am Schluß wird die Trefferquote prozentual ausgegeben.
Nur eines sei hier verraten: Unter 60% ist bei der Bundeswehr kein Rang zu gewinnen.
Bei der grafischen Gestaltung des Bildschirms hat sich der Autor Mühe gegeben und das Extended-Basic gut genutzt. Der Programmierstil zeigt jedoch deutlich, daß der Autor mit den vielfältigen Möglichkeiten dieser Programmiersprache noch nicht ganz zu Rande kommt. Mehrfachbelegungen, kürzere Umwandlungen und die nicht optimale Zeilenausnutzung sind nur einige der Kritikpunkte, von denen wir, unter anderem mit Hilfe des Kompressors, einige gleich ausgebügelt haben. Dafür ließ dieses Programm jedoch jeglichen Fehler in der Programmierung vermissen und läuft absturzsicher.

UNSERE MAILBOX STEHT!

Was lange währt: Unsere Mailbox steht! Unter der Rufnummer 089/183951 können nun alle DFÜ-Freaks unseren Computer anwählen.

WICHTIGE HINWEISE!

**Die Box befindet sich noch im Probe-Betrieb — kleinere Fehler können noch auftreten, werden aber sofort ausgemerzt. Die Telefonnummer ist eine Direktleitung zum Computer, kann also nicht für normale Gespräche genutzt werden!
Die Box ist menü-gesteuert, beachten Sie bitte die Bedienungshinweise!
Die Box ist an Wochentagen zunächst nur von 19 Uhr abends bis 9 Uhr morgens geöffnet, Samstags, Sonntags sowie an Feiertagen selbstverständlich durchgehend von 0 bis 24 Uhr.**

```
1Ø REM  ******************
2Ø REM  *    DIE FRONT    *
3Ø REM  ******************
4Ø REM  *       VON       *
5Ø REM  *  JENS UFERMAN   *
6Ø REM  * NUSSSBAUMWEG 9 *
7Ø REM  *462Ø CASTRROP R.*
8Ø REM  *    (C) 1985     *
9Ø REM  *       BY        *
1ØØ REM *    UFERSOFT     *
11Ø REM *****************
115 !
12Ø CALL DELSPRITE(ALL):: CA
LL CHARSET :: CALL CLEAR ::
FOR C=96 TO 126
15Ø READ C$ :: CALL CHAR(C,C
$)
17Ø NEXT C
18Ø DATA ØØØØØØØØØ1Ø1Ø1Ø3,42
227DFFFFFFFFBF,ØØ6ØFEFFFFFFF
FFF,ØØØØØØØØ8Ø8ØEØF,Ø3ØØØØØ1
Ø3Ø7Ø7Ø7
19Ø DATA 67E7EFFFFFFFFFEFD,FF
FFFFFFFFFFFFFF,FØEØFØFØFØFØF
2FC,ØØØ3Ø3Ø71F1F3F3F,ØØØØ8ØC
ØEØFØFØF9
2ØØ DATA Ø7ØØØØØØØØØØØØØE,83Ø
3Ø3Ø7Ø7ØF1F3F,7E7EBCB8B8BØ8Ø
C,ØØØ3Ø3Ø1ØØØ1Ø7Ø3,7FFFFFFFF
EFEFEFE
21Ø DATA FFFF9F3F7F7F7F7F,CØ
CØCØ8Ø8ØC6FFFF,BE3E3EFFFF3F3
E3E,3F3F1F1F1F1F3F7F,F7F7F7E
FDFBF8
22Ø DATA FFFFFFFFFEFCE,FFFFF
CFØFØFØFØF,CF8F1F3F7CF8FØE,Ø
1Ø3Ø3Ø1,7EFCFCF8,7F3FØFØ7Ø3Ø
3Ø1Ø1
23Ø DATA F8FEFFFF8E8ØCØC,FØF
ØFØFØFØFØF Ø78,Ø3Ø3Ø3Ø1Ø1Ø1,E
ØEØEØFØF8F87,787C3C
24Ø CALL CHAR(35,"Ø"):: CALL
 CHAR(42,"Ø"):: CALL COLOR(1
,16,2,2,2,11,3,16,7):: CALL
CHAR(5Ø,"Ø"):: CALL CHAR(91,
"ØØØØØØ7EØ")
28Ø CALL COLOR(9,16,7,1Ø,16,
7,11,16,7,12,16,7,5,16,2,6,1
6,2,7,16,2,8,16,2)
29Ø CALL HCHAR(1,1,35,256)::
 CALL HCHAR(9,1,5Ø,256):: CA
LL HCHAR(17,1,42,256):: DISP
LAY AT(1Ø,1)SIZE(17):"222222
2222222'abc"
31Ø DISPLAY AT(11,1)SIZE(17)
:"22222222222222defg" :: DISP
LAY AT(12,1)SIZE(17):"222222
22222hijkfl" :: DISPLAY AT(1
3,1)SIZE(17):"2222222222mnof
ffp"
34Ø DISPLAY AT(14,1)SIZE(17)
:"22222222222qrstuv" :: DISP
LAY AT(15,1)SIZE(17):"222222
2222wxyz2(2" :: DISPLAY AT(1
6,1)SIZE(17):"22222222222222|}
2~2" :: DISPLAY AT(4,11)SIZE
(9):"UFER[SOFT"
38Ø DISPLAY AT(6,9)SIZE(13):
"PRAESENTIERT!" :: CALL CHAR
(132,"3C4299A5A5A5A5A5A5A5A5
A5A599423CE7A5B595959595959199
89A9A9A9ADA5E7")
4ØØ CALL CHAR(136,"FF81EF282
82828282828282828282C243C")
41Ø CALL CHAR(128,"FF81BDA5A
7AØBF81BDA7AØAØAØBØ9ØFØFE82B
9A9A5A5A9B286B2A9A5A5B595F7"
)
42Ø CALL COLOR(13,2,11,14,2,
11):: CALL HCHAR(2Ø,13,128):
: CALL HCHAR(21,13,129):: CA
LL HCHAR(2Ø,15,13Ø):: CALL H
CHAR(21,15,131)
43Ø CALL HCHAR(2Ø,17,132)::
CALL HCHAR(21,17,133):: CALL
 HCHAR(2Ø,19,134):: CALL HCH
AR(21,19,135):: CALL HCHAR(2
Ø,21,136):: CALL HCHAR(21,21
,137)
44Ø FOR I=1 TO 3 :: FOR M=1
TO 39 :: CALL COLOR(13,16,11
,14,16,11):: NEXT M
45Ø FOR V=1 TO 3Ø STEP 3 ::
CALL SOUND(-5Ø,11ØØ,V,22ØØ,V
,44ØØ,V):: CALL SOUND(-5Ø,44
ØØ,V,88ØØ,V,132ØØ,V):: CALL
COLOR(13,2,11,14,2,11)
46Ø NEXT V :: NEXT I :: CALL
 CLEAR :: CALL CHARSET :: CA
LL SCREEN(7)
49Ø FOR I=Ø TO 14 :: CALL CO
LOR(I,11,2):: NEXT I :: DISP
LAY AT(12,2)SIZE(29):"ANLEIT
UNG LESEN? J/N  >[E]" :: ACC
EPT AT(12,23)SIZE(1)VALIDATE
("JN")BEEP:Q$ :: IF Q$="N" T
HEN 66Ø ELSE 52Ø
52Ø PRINT "SIE WERDEN ZUR BU
NDESWEHR    EINBERUFEN. HIER
MUESSEN SIEEINEN EIGNUNGSTES
T AM        COMPUTER DURCHFUE
HREN." :: PRINT
53Ø PRINT "DABEI SOLL GEPRUE
FT WERDEN, OB SIE DIE BEFAEH
IGUNG ZUM   GEFREITEN,OFFIZIE
R,          HAUPTMANN,MAJOR O
DER GENERALHABEN." :: PRINT
54Ø PRINT "IHRE AUFGABE IST
ES, AUF DEMBILDSCHIRM DEN NA
CHSCHUB DESGEGNERS ZU UNTERB
```

LISTINGS

```
RECHEN." :: PRINT
55Ø PRINT "DAZU SOLLEN SIE D
IE FEIND-  LICHEN OBJEKTE (F
LUGZEUG,   U-BOOT,LKW,GELEIT
ZUG UND    PANZER) MIT WENIG
EN SCHUES- SEN AUSSCHALTEN."
 :: PRINT
57Ø PRINT "     > TASTE DRUE
CKEN <" :: CALL MUSIK :: CAL
L CLEAR :: PRINT "TAKTISCHE
FAEHIGKEITEN UND  REAKTIONSV
ERMOEGEN SIND     HIERBEI EN
TSCHEIDEND."
61Ø PRINT :: PRINT "
 NUR MUT!            " :: PRIN
T :: PRINT "=>    KIMME,KORN,
SCHUSS    <=" :: PRINT :: PRI
NT :: PRINT "         VIEL FRE
UDE!!" :: PRINT :: PRINT "
    > TASTE DRUECKEN <" :: CA
LL MUSIK
66Ø CALL DELSPRITE(ALL):: CA
LL SCREEN(7):: P=Ø :: S=Ø ::
 CALL CLEAR :: CALL CHARSET
:: DISPLAY AT(2,6):"SCHWIERI
GKEITSGRADE:          ========
=============="
661 FOR I=1 TO 8 :: CALL COL
OR(I,2,7):: NEXT I :: DISPLA
Y AT(11,2):"1 = GEFREITER" :
: DISPLAY AT(13,2):"2 = OFFI
ZIER" :: DISPLAY AT(15,2):"3
 = HAUPTMANN" :: DISPLAY AT(
17,2):"4 = MAJOR"
71Ø DISPLAY AT(19,2):"5 = GE
NERAL" :: DISPLAY AT(22,1):"
WAS MOECHTEST DU SEIN?  >[E]
" :: ACCEPT AT(22,24)SIZE(1)
VALIDATE("12345")BEEP:R
73Ø DISPLAY AT(24,1):"WIEVIE
L SCHUSS? <=99      >[E]" :: A
CCEPT AT(24,23)SIZE(2)VALIDA
TE("1234567890")BEEP:S :: SS
=S :: IF R=1 THEN B=1 :: IF
R=2 THEN B=2 :: IF R=3 THEN
B=3 :: IF R=4 THEN B=4 :: IF
 R=5 THEN B=5
75Ø CALL SCREEN(15):: CALL C
HAR(128,"ØØØØ4121117Ø913157F1
513Ø9172141ØØØØØØØ4Ø8DØ2Ø9Ø5
ØFC5Ø9Ø2ØDØØ8Ø4Ø")
77Ø REM U-BOOT
78Ø CALL CHAR(132,"ØØØØØØØØØ
ØØØØCØ43F7FFFØØØØØØØØØØØØØØØØ
ØØØØØ7Ø7Ø7ØFCFEFFØ")
79Ø REM FLUGZEUG
8ØØ CALL CHAR(136,"ØØØØØØØØØ
ØØE91BFFØBF87ØØØØØØØØØØØØØØØØ
ØØØØ1Ø181E37FF8EØ")
81Ø REM ZUG
82Ø CALL CHAR(14Ø,"ØØØØ18181
819191F3F7F7F7F7F4Ø5B1BØØØØØ
ØØØ7C4444C4FCFCFCFDFFØ16C6C"
)
83Ø REM TIEFLADER
84Ø CALL CHAR(44,"ØØØØØØØØØØ
ØØØEØFØE133ØAAFF8Ø3636ØØØØØØ
ØØØØØØØØEØØØFØ14A5FFØ1D8D8")
85Ø REM LKW
86Ø CALL CHAR(1Ø8,"ØØØØØØØØØ
Ø3D25257D7D7FØØ1818ØØØØØØØØØ
ØØØØØF8F8F8F8F8FCØØ3Ø3Ø")
87Ø REM PANZER
88Ø CALL CHAR(124,"ØØØØØØØØØ
ØØØØØØØØØØ1FFØ31FØØ152AØØØØØ
ØØØØØØØØØØØØFØD89FFFFCØØ54AA"
):: CALL CLEAR
9ØØ CALL CHAR(88,"FFFFFFFFFF
FFFFFF",89,"8ØEØEØFØFØFCFEFF
",9Ø,"Ø1Ø7Ø7ØFØF3F7FFF",91,"
Ø1Ø2Ø4Ø81Ø2Ø4Ø8Ø",92,"8Ø4Ø2Ø
1ØØ8Ø4Ø2Ø1")
91Ø CALL CHAR(93,"ØØØØØØØØØØ
ØØØØFF"):: CALL CHAR(94,"FFØ
4Ø81Ø2Ø4Ø8ØFF"):: CALL COLOR
(8,2,15)
94Ø REM
95Ø CALL CHAR(96,"FFFFFFFFFF
FFFFFF",97,"FFFEFCF8FØEØCØ8Ø
",98,"FF7F3F1FØFØ7Ø3Ø1':: C
ALL COLOR(9,4,8)
96Ø REM
97Ø CALL CHAR(1Ø4,"FFFFFFFFF
FFFFFFF",1Ø5,"FF7F3F1FØFØ7Ø3
Ø1",1Ø6,"FFFEFCF8FØEØCØ8Ø"):
: CALL COLOR(1Ø,3,8)
98Ø CALL CHAR(1ØØ,"FFE3CØCØ8
1ØØØØ8Ø":: CALL CHAR(1Ø1,"C
3F3F3F3F3F3C18Ø")
99Ø REM
1ØØØ CALL CHAR(112,"FFFFFFFF
FFFFFFFF",113,"FF7F3F1FØFØ7Ø
3Ø1",114,"FFFEFCF8FØEØCØ8Ø")
:: CALL COLOR(11,13,5):: CAL
L CHAR(116,"E7C381ØØA581A585
")
1Ø2Ø CALL CHAR(115,"Ø1Ø3Ø7ØF
1F3F7FFF")
1Ø3Ø REM
1Ø4Ø CALL CHAR(12Ø,"FFFFFFFF
FFFFFFFF"):: CALL COLOR(12,5
,5)
1Ø5Ø REM
1Ø6Ø CALL CHAR(4Ø,"FFFFFFFFF
FFFFFFF"):: CALL COLOR(2,8,8
):: CALL CHAR(99,"Ø1Ø3Ø7ØF1F
3F7FFF)
1Ø8Ø REM
1Ø9Ø CALL COLOR(1,15,15):: D
ISPLAY AT(2,1):"    ZY     ZY
   ZY              " :: DISPLAY
```

```
AT(3,1):" ZXXXY  ZXXYZYZXXY
ZYZY    Z" :: DISPLAY AT(4,
1):"XXXXXXXXXXXXXXXXXXXXXXXY
ZYZX"
113Ø DISPLAY AT(5,1):"XXXXXX
XXXXXXXXXXXXXXXXXXXXXX" :: D
ISPLAY AT(6,1):"pppppppppppp
pppppppppppppppp" :: DISPLAY
 AT(7,1):"ppppppppppppppppppp
ppppppppr"
116Ø DISPLAY AT(8,1):"pppppp
ppppppppppprxxxxxxxxxs" :: D
ISPLAY AT(9,1):"pppppppppppp
ppprxxsppppppppp" :: DISPLAY
 AT(1Ø,1):"pppppppppppppprxx
sppppppppp"
119Ø DISPLAY AT(11,1):"^^^^^
^^^^ ^^^^^^^^^^^^^^^ ^^ " ::
DISPLAY AT(12,1):"hhhhhhhhhh
hhhhhhhhhhhhhhhhhh" :: DISPL
AY AT(13,1)"hhhhhhhhhhhj((i
hhhhhhhhhhhhh"
122Ø DISPLAY AT(14,1):'''''
''''' a(((c''''''''''" ::
DISPLAY AT(15,1):''''''a(
(((c ''''''''''''''" :: DISPL
AY AT(16,1):'''a((((((c''
'''''''''''''''"
125Ø DISPLAY AT(17,1):'((((
((((((c''''''''''''''" ::
DISPLAY AT(18,1):"(((((((c''
''''''''''''''''''" :: DISPL
AY AT(19,1):"''''''''''''''
'''''''''''''''"
128Ø DISPLAY AT(2Ø,1):'''''
'''''''''''''''''''''''''"
129Ø FOR I=21 TO 24 :: DISPL
AY AT(I,1):"XXXXXXXXXXXXXXXX
XXXXXXXXXXXX" :: NEXT I
13ØØ CALL VCHAR(21,1,88,4)::
 CALL VCHAR(21,2,88,4):: CAL
L VCHAR(21,31,88,4):: CALL V
CHAR(21,32,88,4)
131Ø CALL HCHAR(19,1,96,64):
: CALL VCHAR(14,1,96,3):: CA
LL VCHAR(14,2,96,3):: CALL V
CHAR(14,31,96,7):: CALL VCHA
R(14,32,96,7)
132Ø CALL VCHAR(17,1,4Ø,2)::
 CALL VCHAR(17,2,4Ø,2)
133Ø CALL HCHAR(11,1,94,32):
: CALL HCHAR(12,1,1Ø4,32)::
CALL HCHAR(13,1,1Ø4,2):: CAL
L HCHAR(13,31,1Ø4,2)
134Ø CALL VCHAR(6,1,112,5)::
 CALL VCHAR(6,2,112,5):: CAL
L VCHAR(6,31,112,5):: CALL V
CHAR(6,32,112,5):: CALL HCHA
R(7,31,12Ø,2)
135Ø CALL HCHAR(14,1Ø,1ØØ)::
 CALL HCHAR(15,1Ø,1Ø1):: CAL
L VCHAR(3,1,88,3):: CALL HCH
AR(4,2,88,2):: CALL VCHAR(2,
1,89):: CALL HCHAR(3,2,89)
137Ø CALL VCHAR(3,31,88,3)::
 CALL VCHAR(3,32,88,3):: CAL
L VCHAR(2,31,9Ø):: CALL HCHA
R(2,32,89)
138Ø CALL HCHAR(15,29,1ØØ,2)
:: CALL HCHAR(16,29,1Ø1,2)::
 CALL HCHAR(16,19,1Ø1):: CAL
L HCHAR(15,19,1ØØ)
139Ø CALL HCHAR(18,2Ø,1ØØ,2)
:: CALL HCHAR(19,2Ø,1Ø1,2)::
 CALL HCHAR(18,26,1ØØ):: CAL
L HCHAR(19,26,1Ø1)
14ØØ CALL HCHAR(5,2,88):: CA
LL HCHAR(7,5,116,2):: CALL H
CHAR(6,1Ø,116):: CALL HCHAR(
7,11,116):: CALL HCHAR(8,8,1
16):: CALL HCHAR(8,2,116)
141Ø FOR M=3 TO 7 :: CALL CO
LOR(M,7,2):: NEXT M :: CALL
SPRITE(#1,128,16,136,112)
143Ø CALL MAGNIFY(3)
144Ø CALL COLOR(1,2,15,8,2,1
5):: CALL SCREEN(15):: Z=Ø :
: CALL DELSPRITE(#8,#2,#3,#4
,#5,#6,#7)
145Ø REM PROGRAMM
146Ø REM
147Ø DISPLAY AT(22,2)SIZE(13
):"OBJEKT:";A$ :: DISPLAY AT
(23,2)SIZE(13):"PUNKTE:";P :
: CALL COLOR(1,2,15,8,2,15):
: DISPLAY AT(22,19)SIZE(1Ø):
"SCHUSS:";S
151Ø FOR Q=1 TO 1Ø :: CALL J
OYST(1,X1,Y1):: CALL MOTION(
#1,-4*Y1,4*X1): NEXT Q :: R
ANDOMIZE :: J=INT(RND*6)
154Ø IF J=Ø THEN 2Ø8Ø :: IF
J=1 THEN 155Ø :: IF J=2 THEN
 169Ø :: IF J=3 THEN 182Ø ::
 IF J=4 THEN 195Ø :: IF J=5
THEN 219Ø
155Ø Z=8 :: Z=Z+R :: CALL SP
RITE(#2,14Ø,2,72,24Ø,Ø,-Z,#7
,44,2,72,255,Ø,-Z):: DISPLAY
 AT(22,2)SIZE(13):"OBJEKT:LO
CKXX"
157Ø CALL JOYST(1,X1,Y1):: C
ALL MOTION(#1,-4*Y1,4*X1)::
CALL SOUND(1ØØ,-5,Ø,11Ø,Ø)::
 CALL POSITION(#2,X,Y):: IF
Y<9 THEN 144Ø
159Ø CALL KEY(1,K,N):: IF N=
Ø THEN 157Ø
16ØØ DISPLAY AT(22,19)SIZE(1
Ø):"SCHUSS:";S :: CALL SOUND
(1ØØ,-6,Ø): S=S-1 :: IF S<=
```

LISTINGS

```
Ø THEN 231Ø
161Ø CALL COINC(#1,#2,3,A)::
 IF A=-1 THEN 162Ø ELSE 164Ø
162Ø CALL DELSPRITE(#2):: CA
LL COLOR(1,7,7,8,2,7):: FOR
L=Ø TO 16 :: CALL SOUND(-99,
-7,L,12Ø,L,131,L):: CALL SCR
EEN(7):: NEXT L
163Ø CALL SCREEN(11):: CALL
COLOR(1,11,11,8,2,11):: P=P+
15Ø :: GOTO 144Ø
164Ø CALL COINC(#1,#7,3,A2):
: IF A2=-1 THEN 165Ø ELSE 16
7Ø
165Ø CALL DELSPRITE(#7):: CA
LL COLOR(1,7,7,8,2,7):: FOR
L=Ø TO 16 :: CALL SOUND(1ØØ,
-7,L,12Ø,L,131,L):: CALL SCR
EEN(7):: NEXT L
166Ø CALL SCREEN(11):: CALL
COLOR(1,11,11,8,2,11):: P=P+
1ØØ :: GOTO 144Ø
167Ø CALL SOUND(1ØØ,-5,Ø,11Ø
,Ø):: CALL POSITION(#2,X,Y):
: IF Y<9 THEN 144Ø ELSE 159Ø
169Ø REM
17ØØ REM FLUGZEUG
171Ø Z=1Ø :: Z=Z+R :: CALL S
PRITE(#3,136,16,23,24Ø,1,-Z)
:: DISPLAY AT(22,2)SIZE(13):
"OBJEKT:FLUGXX"
173Ø CALL JOYST(1,X1,Y1):: C
ALL MOTION(#1,-4*Y1,4*X1)::
CALL SOUND(1ØØ,-7,Ø,11Ø,Ø)::
 CALL POSITION(#3,X,Y):: IF
Y<9 THEN 144Ø
175Ø CALL KEY(1,K,N):: IF N=
Ø THEN 173Ø
176Ø DISPLAY AT(22,19)SIZE(1
Ø):"SCHUSS:";S :: CALL SOUND
(1ØØ,-6,Ø):: S=S-1 :: IF S<=
Ø THEN 231Ø
177Ø CALL COINC(#1,#3,3,B)::
 IF B=-1 THEN 178Ø ELSE 18ØØ
178Ø CALL DELSPRITE(#3):: CA
LL COLOR(1,7,7,8,2,7):: FOR
L=Ø TO 16 :: CALL SOUND(1ØØ,
-7,L,12Ø,L,131,L):: CALL SCR
EEN(7):: NEXT L
179Ø CALL SCREEN(11):: CALL
COLOR(1,11,11,8,2,11):: P=P+
2ØØ :: GOTO 144Ø
18ØØ CALL SOUND(1ØØ,-7,Ø,11Ø
,Ø):: CALL POSITION(#3,X,Y):
: IF Y<9 THEN 144Ø ELSE 175Ø
182Ø REM
183Ø REM PANZER
184Ø Z=4 :: Z=Z+R :: CALL SP
RITE(#4,124,2,112,216,Ø,-Z):
: DISPLAY AT(22,2)SIZE(13):"
OBJEKT:PANXXX"
186Ø CALL JOYST(1,X1,Y1):: C
ALL MOTION(#1,-4*Y1,4*X1)::
CALL SOUND(1ØØ,-8,Ø,11Ø,8)::
 CALL POSITION(#4,X,Y):: IF
Y<158 THEN 144Ø
188Ø CALL KEY(1,K,N):: IF N=
Ø THEN 186Ø
189Ø DISPLAY AT(22,19)SIZE(1
Ø):"SCHUSS:";S :: CALL SOUND
(1ØØ,-6,Ø):: S=S-1 :: IF S<=
Ø THEN 231Ø
19ØØ CALL COINC(#1,#4,3,C)::
 IF C=-1 THEN 191Ø ELSE 193Ø
191Ø CALL DELSPRITE(#4):: CA
LL COLOR(1,7,7,8,2,7):: FOR
L=Ø TO 16 :: CALL SOUND(1ØØ,
-7,L,12Ø,L,131,L):: CALL SCR
EEN(7):: NEXT L
192Ø CALL SCREEN(11):: CALL
COLOR(8,2,11,1,11,11):: P=P+
1ØØ :: GOTO 144Ø
193Ø CALL SOUND(1ØØ,-8,Ø,11Ø
,8):: CALL POSITION(#4,X,Y):
: IF Y<158 THEN 144Ø ELSE 18
8Ø
195Ø REM
196Ø REM U-BOOT
197Ø Z=1 :: Z=Z+R :: CALL SP
RITE(#5,132,2,51,216,Ø,-Z)::
 DISPLAY AT(22,2)SIZE(13):"O
BJEKT:UXBOOT"
199Ø CALL JOYST(1,X1,Y1):: C
ALL MOTION(#1,-4*Y1,4*X1)::
CALL SOUND(1ØØ,-4,9):: CALL
POSITION(#5,X,Y):: IF Y<176
THEN 144Ø
2Ø1Ø CALL KEY(1,K,N):: IF N=
Ø THEN 199Ø
2Ø2Ø DISPLAY AT(22,19)SIZE(1
Ø):"SCHUSS:";S :: CALL SOUND
(1ØØ,-6,Ø):: S=S-1 :: IF S<=
Ø THEN 231Ø
2Ø3Ø CALL COINC(#1,#5,3,V)::
 IF V=-1 THEN 2Ø4Ø ELSE 2Ø6Ø
2Ø4Ø CALL DELSPRITE(#5):: CA
LL COLOR(1,7,7,8,2,7):: FOR
L=Ø TO 16 :: CALL SOUND(1ØØ,
-7,L,12Ø,L,131,L):: CALL SCR
EEN(7):: NEXT L
2Ø5Ø CALL COLOR(1,11,11,8,2,
11):: P=P+5Ø :: GOTO 144Ø
2Ø6Ø CALL SOUND(1ØØ,-4,Ø)::
CALL POSITION(#5,X,Y):: IF Y
<176 THEN 144Ø ELSE 2Ø1Ø
2Ø8Ø Z=3 :: Z=Z+R :: CALL SP
RITE(#6,124,2,136,192,Ø,-Z):
: DISPLAY AT(22,2)SIZE(13):"
OBJEKT:PANXXX"
21ØØ CALL JOYST(1,X1,Y1):: C
ALL MOTION(#1,-4*Y1,4*X1)::
```

```
CALL SOUND(1ØØ,-8,Ø,11Ø,8)::
 CALL POSITION(#6,X,Y):: IF
Y<168 THEN 144Ø
212Ø CALL KEY(1,K,N):: IF N=
Ø THEN 21ØØ
213Ø DISPLAY AT(22,19)SIZE(1
Ø):"SCHUSS:";S :: CALL SOUND
(1ØØ,-6,Ø): S=S-1 :: IF S<=
Ø THEN 231Ø
214Ø CALL COINC(#6,#1,3,U)::
 IF U=-1 THEN 215Ø ELSE 217Ø
215Ø CALL DELSPRITE(#6):: CA
LL COLOR(1,7,7,8,2,7):: FOR
L=Ø TO 16 :: CALL SOUND(1ØØ,
-7,L,12Ø,L,131,L):: CALL SCR
EEN(7):: NEXT L
216Ø CALL SCREEN(11):: CALL
COLOR(1,11,11,8,2,11):: P=P+
25Ø :: GOTO 144Ø
217Ø CALL SOUND(1ØØ,-8,Ø,11Ø
,8):: CALL POSITION(#6,X,Y):
: IF Y<168 THEN 144Ø ELSE 21
2Ø
219Ø REM LKW
22ØØ Z=2.6 :: Z=Z+R :: CALL
SPRITE(#8,1Ø8,9,1Ø4,56,Ø,-Z)
:: DISPLAY AT(22,2)SIZE(13):
"OBJEKT:LKWXXX"
222Ø CALL JOYST(1,X1,Y1):: C
ALL MOTION(#1,-4*Y1,4*X1)::
CALL SOUND(1ØØ,-5,4,11Ø,Ø)::
 CALL POSITION(#8,X,Y):: IF
Y<7 THEN 144Ø
224Ø CALL KEY(1,K,N):: IF N=
Ø THEN 222Ø
225Ø DISPLAY AT(22,19)SIZE(1
Ø):"SCHUSS:";S :: CALL SOUND
(1ØØ,-6,Ø):: S=S-1 :: IF S<=
Ø THEN 155Ø
226Ø CALL COINC(#1,#8,3,O)::
 IF O=-1 THEN 227Ø ELSE 229Ø
227Ø CALL DELSPRITE(#8):: CA
LL COLOR(1,7,7,8,2,7):: FOR
L=Ø TO 16 :: CALL SOUND(1ØØ,
-6,L,12Ø,L,131,L):: CALL SCR
EEN(7):: NEXT L
228Ø CALL SCREEN(11):: CALL
COLOR(1,11,11,8,2,11):: P=P+
15Ø :: GOTO 144Ø
229Ø CALL SOUND(1ØØ,-5,4,11Ø
,Ø):: CALL POSITION(#8,X,Y):
: IF Y<7 THEN 144Ø ELSE 224Ø
231Ø FOR M=1 TO 4ØØ :: NEXT
M :: CALL CHARSET :: CALL DE
LSPRITE(ALL):: CALL CLEAR ::
 CALL SCREEN(2):: FOR I=1 TO
 8 :: CALL COLOR(I,16,2):: N
EXT I :: Q=P/SS :: DISPLAY A
T(2,Ø):"IHRE PUNKTZAHL:";P
2322 DISPLAY AT(4,Ø):"IHR LE
ISTUNGSQUOTIENT" :: DISPLAY
AT(5,Ø):"      (PUNKTE/SCHUSS
):";Q :: DISPLAY AT(8,Ø):">=
 6Ø BEFAEHIGUNG ERFUELLT"
2325 DISPLAY AT(9,Ø):"<=  6Ø
 SIE WERDEN WEGEN UN-" :: DI
SPLAY AT(1Ø,Ø):"        HAEHI
GKAEIT EIN JAHR" :: DISPLAY
AT(11,Ø:"       ZURUECKGEST
ELLT."
2328 DISPLAY AT(15,Ø):"NOCH
EINEN TEST? (J/N)" :: CALL M
AGNIFY(4):: CALL SPRITE(#1,1
24,16,16Ø,192,Ø,-5,#2,136,16
,23,24Ø,1,-8)
233Ø CALL KEY(3,K,S):: IF K=
74 OR K=1Ø6 THEN 66Ø ELSE IF
 S=Ø THEN 233Ø ELSE IF K=78
OR K=11Ø THEN CALL CLEAR
235Ø END
236Ø SUB MUSIK
237Ø RESTORE 241Ø :: FOR LOO
P=1 TO 44 :: READ T,D :: CAL
L SOUND(D*1.5,T,9):: CALL KE
Y(3,K,PV):: IF K>-1 THEN SUB
EXIT
24ØØ NEXT LOOP :: GOTO 237Ø
241Ø DATA 391,125,44Ø,125,49
3,25Ø,493,25Ø,44Ø,125,391,12
5,44Ø,25Ø,44Ø,25Ø
242Ø DATA 391,125,44Ø,125,49
3,125,523,125,587,25Ø,523,12
5,493,125,44Ø,25Ø,25ØØØ,25Ø
243Ø DATA 587,125,659,125,58
7,125,523,125,44Ø,25Ø,44Ø,12
5,44Ø,125
244Ø DATA 493,25Ø,587,25Ø,25
ØØØ,4,587,125,659,125,587,12
5,523,125,44Ø,25Ø,44Ø,125,44
Ø,125,493,25Ø
245Ø DATA 587,25Ø,659,125,52
3,125,493,125,587,125,493,25
Ø,44Ø,125,391,125,391,25Ø,25
ØØØ,25Ø
246Ø SUBEND
```

UNSERE MAIBOX STEHT!

Was lange währt: Unsere Mailbox steht! Unter der Rufnummer 089/183951 können nun alle DFÜ-Freaks unseren Computer anwählen.

WICHTIGE HINWEISE!

Die Box befindet sich noch im Probe-Betrieb – kleinere Fehler können noch auftreten, werden aber sofort ausgemerzt. Die Telefonnummer ist eine Direktleitung zum Computer, kann also nicht für normale Gespräche genutzt werden!
Die Box ist menü-gesteuert, beachten Sie bitte die Bedienungshinweise!
Die Box ist an Wochentagen zunächst nur von 19 Uhr abends bis 9 Uhr morgens geöffnet, Samstags, Sonntags sowie an Feiertagen selbstverständlich durchgehend von 0 bis 24 Uhr.

EXTERNER CONTROLLER MIT DOUBLE SIDE

Ist es möglich, den externen Controller der Firma Texas Instruments über einen Aufrüstsatz für doppelseitige Laufwerke zu verwenden? Ist es, falls ein solcher nicht existiert, möglich, dem externen Controller das Betriebssystem des internen einzubauen, wenn ja, wie?
Joachim Stulz, Lahr

Der externe Controller ist nicht fähig, doppelseitig zu schreiben und zu lesen. Es fehlt ihm dazu die komplette Ansteuerung für doppelseitige Laufwerke. Ebenfalls ist er vom Aufbau her von Grund auf verschieden zum internen Controller. Ein Austausch der Betriebssysteme ist daher nicht möglich.

BUSINESS GRAPH 99

Wie wir inzwischen von mehreren Lesern und der Firma 'TRITON' USA erfahren haben, nimmt diese keine Lieferung außerhalb des amerikanischen Kontinents an. Es ist daher, da noch kein deutscher Händler dieses Programm vertreibt, in Deutschland nicht zu erwerben. Sollte es inzwischen jedoch eine Quelle geben, so kann man uns diese jederzeit mitteilen. Wir werden sie dann in einer der nächsten TI-REVUEs veröffentlichen.

MANIPULATION DES SPIELS LABYRINTH OF OX

Das Adventure läßt sich auf folgende Art manipulieren: Nachdem man die drei Gegenstände im oberen Teil der Höhle eingesammelt hat, durchbreche man eine der beiden Seitenwände und lasse den Bildschirm nach oben scrollen, bis man den Höhlenausgang sieht. Jedes der Zeichen außerhalb des Rands wird vom Computer als eingesammelter Gegenstand gewertet. Auf der Höhe des Ausgangs kann man wieder die Seitenwand durchbrechen und die Höhle betreten. Weiterhin habe ich eine Eingabehilfe der Zeichen CTRL-A bis CTRL-O entwickelt. Nach Eingabe und Ausführung der Zeile
FOR A=97 TO 111 :: CALL CHARPAT(A,AS) :: CALLCHAR(A+32, SEGS(AS,5,12)+"007C") :: NEXT A
sind sie als unterstrichene Buchstaben sichtbar. Dies ist jedoch nur in X-Basic möglich.
Jürgen Tölke, Emmendingen

Vielen Dank für Ihren Hinweis. Er wird für viele User sehr hilfreich sein.

CLUB-ECKE

TI-CLUB BAUNATAL — DER CLUB MIT DEM GUTEN SERVICE!

Vor zwei Jahren, genau im Januar 1985, erschien unser erstes Clubmagazin, und niemand hätte damals geahnt, daß aus dem ehemaligen TI-Club einmal der große TI-Club Baunatal werden würde. In letzter Zeit haben wir wieder sehr viele Anfragen an den Club, so daß wir meinen, eine erneute Vorstellung unserer Angebotspalette wäre angebracht.
Interessant ist unser „USA-Service" durch den Sie die Möglichkeit haben, aus den Staaten Hard- und Software zu importieren. Die Arbeit erledigen wir für den Besteller, Sie brauchen lediglich aus dem USA-Katalog Ihre gewünschte Hard- oder Software auszusuchen und uns dies mitzuteilen. Der Rest läuft wie von selbst . . .
Des weiteren haben wir noch den TCB-Bestellservice, der in dieser Form wie es ihn jetzt gibt, vollkommen neu ist!
Wer Hard- bzw. Software aus deutschen Landen sucht, ob gebraucht oder neu, teilt uns dies mit. Wir versuchen nun das entsprechende Teil möglichst günstig von einem Händler bzw. von einer Privatperson zu kaufen. Die angeforderte Ware wird dann sofort vom Club an den Interessenten weitergeleitet. Zur Zeit sind sehr viele Module lieferbar, daher lohnt ein Anruf von Zeit zu Zeit immer!
Wer Hardware oder Software zu verkaufen hat, ist bei uns auch willkommen,

INFO KOSTENLOS

denn wir vermitteln gerne Verkäufer und Käufer. Informieren Sie sich über den TCB-Bestellservice, denn Infos sind bei uns vollkommen kostenlos!
Als ganz neuer Service bietet der TCB seit kurzem die DFÜ-Leitung, über die Sie per Akustikkoppler Programme für Ihren 99er bekommen können. Kurzer Anruf bei uns genügt, und Ihr „Wunschprogramm", natürlich keine Copyright geschützten Programme, wird Ihnen „rüber-genudelt"!!!
Schreiben Sie uns, oder rufen Sie uns an. Herrn Orf erreichen Sie jeden Tag von 16.00 Uhr bis 21.30 Uhr unter 0561/491 10 78. Herrn Göbel in Großalmerode erreichen Sie täglich bis 22 Uhr unter 05604/69 99 oder 05602/20 59. Bei Problemen, Fragen, Kritiken scheuen Sie nicht uns anzurufen, denn wir sind nur soweit wie der Griff zum Telefon.
TI-Club Baunatal
Matthias Orf
Birkenallee 34
3507 Baunatal 1
Deutschland

SAAR UND PFALZ — CLUBGRÜNDUNG

Seit etwa Dezember '86 laufen die Vorbereitungen zur Bildung einer TI-99-User-Gruppe im Raum Saarland/Westpfalz. Am Anfang waren wir drei begeisterte TI-User. Wir sprachen alle uns bekannten weiteren Besitzer eines Ti-99 auf unsere Absicht hin an und bekamen von fast allen ein positives Echo.
Zum Thema Programmiersprachen wollen wir kleinere Gruppen bilden, die je nach Interesse in den verschiedenen Programmiersprachen des TI arbeiten sollen.
Die Gruppe soll aber nicht nur eine Tausch- und Programmiergemeinschaft sein, sondern es soll auch mit anderen Clubs zusammengearbeitet werden, sowohl auf der Hardware- als auch auf der Softwareseite.
Eine Freeware-Bibliothek soll ebenfalls jedem User zugänglich gemacht werden für einen Selbstkostenbeitrag von 5,– DM.
So, das wär's fürs erste... wir hoffen, daß auch die letzten TI-User aus dem Raum Saarland/Westpfalz sich bei uns melden.
Unsere Kontaktadresse:
Peter Rieger
Talstraße 64
6750 Kaiserslautern
Tel.: 0631/7 36 69

TI REVUE

Das Magazin für TI 99-4A

Nr. 4/87

Einführung in GPL (V)

WICHTIGER HINWEIS!
Ein Hinweis für die TI-REVUE-Abonnenten! Dies ist die Nummer 4/87 der TI-REVUE!, aber die Nummer 6 der HCA-MSX-REVUE. Und so wird auch bei uns in der Abo-Verwaltung gezählt!

IM TEST:

Speicherprogramierbare Steuerung

DIALOG

Was tun, wenn der Ti-streikt?
Was kaufen: gebrauchte Peri-Box oder neues CPS99?

DAS HEFT IM HEFT FÜR TI-USER

EINFACH HERAUS-TRENNEN

Grüß Gott -

Wie das Leben so spielt: Unlängst erreichten die Redaktion gleich mehrere Anrufe, alle zum gleichen Thema: der Integration der TI REVUE in eine andere Zeitschrift. Nr 1 kam von einem nicht ganz unbekannten rheinischem TI-Club. Der Anrufer zeigte sich erbost, daß mindestens zwei Ausgaben lang nichts über „seinen" Club in der TI REVUE gestanden habe, aber die Zeitschrift XY dafür regelmäßig berichte. Hektische Aktivität innerhalb der TI-Crew. Ergebnis: Das letzte Stück Papier, das die TI REVUE von exakt diesem Club bekam, war mehr als sechs Monate alt. Stattdessen hatte der gleiche Club (oder eines seiner Vorstandsmitglieder?) eben die Zeitschrift XY offensichtlich als seine Hauspostille erkoren und berichtet dort munter über seine Aktivitäten.

Kein Neid den Kollegen gegenüber, nur: Bevor der Sack geschlagen wird, sollte man sich als Clubmitglied den Esel ansehen, der in dem Sack steckt. Heißt: Wenn ein Club nichts liefert, kann auch nichts in der TI REVUE stehen. So einfach ist das.

Die Anrufer 2 — 5 schienen uns dagegen für geistig etwas unterbemittelt zu halten, nicht einmal einer einfachen Zählung für fähig. Das Blatt YZ (bzw. XYZ) bringe, so der Tenor, schon lange mehr über den TI als wir. Weshalb wir sofort ins Archiv stiegen. Ergebnis unserer Zählung (Durchschnitt der jeweils sechs letzten Hefte der beiden genannten Zeitschriften): 6,5 Seiten bei der einen, 13,4 Seiten TI bei der anderen Zeitschrift. Wir wissen nicht, ob das „mehr über den TI ist", Adam Riese scheint in der Bundesrepublik offensichtlich mittlerweile für einen Ausländer gehalten zu werden, wir haben das mindestens 3fache.

Deswegen — und nun wirklich letztmalig: Noch immer ist dieser Titel die Zeitschrift mit den meisten Informationen — auch vom Umfang her — für die noch verbliebenen TI-User. Und wir garantieren — noch immer —, daß dies so bleiben wird.

Ihre TI-Crew

IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Geschäftsführer: Werner E. Seibt
Redaktion: Alfons Bauer, Thomas Dehn, Jürgen Puschmann, Uwe Schröder
Verantwortlich für den Inhalt:
MSX-Teil: Werner E. Seibt
TI-Teil: Oliver Huber (Technik) Hartmut König (Listings)
Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d
Alle: Postfach 1161
8044 Lohhof.
Anfragen bitte nur schriftlich
Druck: Dierichs, Kassel
Es gilt die Honorarliste des Verlages.
Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.
Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

© by CA Verlag, Heßstraße 90, 8000 München 40 SPS und Autoren

Erstverkaufstag: 20.5.1987

TI99/4A

PREISSENKUNG BEI VIELEN ARTIKELN !!

Peripherie

RS 232 Karte (Atronic) 249,–
32 K-Karte (Atronic) 249,–
Compact Peripherie System CPS 99 mit 1 Diskettenlaufwerk DSDD + Statistikmodul 999,–
CPS 99 mit 2 Diskettenlaufwerken DSDD + Statistikmodul 1329,–
Diskettenlaufwerk DSDD mit Einbausatz für Periph.box 349,–
2 Diskettenlaufwerke DSDD mit Einbau- und Kabelsatz für P.box 729,–
Externe 32 K-Erweiterung batteriegepuffert 149,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle 249,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle + Druckerkabel + Epsondrucker LX 800 889,–
dto. + Epsondrucker FX 800 1319,–
dto. + Epsondrucker EX 800 1679,–
dto. + Stardrucker NL-10 1049,–
dto. + Stardrucker NG-10 1069,–
80-Zeichen-Karte (Mechatronic) 449,–
Ext. V-24-Schnittstelle (2 x RS 232) mit druchgeschl. Bus 199,–
MBx-Sprachsteuereinheit + Baseballmodul anschlußfertig 299,–
WIPO Mouse 99 199,–
RGB-Modulator mit Scartstecker 179,–
Akustikkoppler Dataphon S 21 d + externe V-24-Schnittstelle + Verbindungskabel 499,–
Akustikkoppler Dataphon S 21 d + Kabel für RS 232 Karte 279,–
Joystickinterface + 2 Joysticks Quickshot II 85,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter) 115,–
Modulexpander 3-fach 125,–

Modulsoftware

Extended Basic II Plus 229,–
Mini Memory 169,–
Mini Memory + TMS 9900 Assemblerhandbuch Mini Memory dt. 225,–
Editor/Assembler (32 K notw.) 159,–
TI-Writer (32 K notw.) 199,–
Multiplan (32 K notw.) 199,–
TI-Logo II (32 K notw.) 199,–
Diskfixer (Navarone) 109,–
Terminal Emulator II 85,–
Connect four, Attack je 19,–
Alpiner, Chisholm Trail, Othello, Car Wars, Tomstone City je 29,–
Defender, Dig Dug, Invaders, Munchman, Moonmine, Statistik je 39,–
Fathom, Jungle Hunt, Moon Patrol, Hopper, Parsec, Indoore Soccer je 49,–
Congo Bongo, Burgertime, Espial, Moonsweeper, Microsurgeon, Treasure Island, Donkey Kong, Tunnels of doom, Protector II, Shamus je 59,–
Adenvturemodul, Buck Rogers, Return to Pirates Isle, Star Trek, Video Chess, Popeye je 69,–
Pole Position, Miner 2049 je 79,–
TI-Artist (Diskette) 99,–
Turbo Pascal (Diskette) 149,–

Bücher

Editor/Assembler Handbuch dt. 98,–
TMS 9900 Assemblerhandbuch für das Mini Memory dt. 78,–
TI-99/4 A Intern dt. 38,–
Assembler-Kurs (99er-Journal) 59,–
Introduction to Assembly Language (englsich) 49,–
Grafiktablett Supersketch + Defernder + Dig Dug + Donkey Kong + Statistik 199,–

Commodore / Schneider

Plus 4 + Floppy-Disk VC 1551 499,–
Commodore SC-64 1479,–
Commodore C 128 D 1149,–
Amiga 500 1199,–
Amiga 2000 2599,–
Schn. CPC 6128 Grünmonitor 729,–
Schn. CPC 6128 Farbmonitor 1189,–
Schneider PC 1512 mit Schwarzweißmonitor + 1 Laufwerk 1349,–
dto. + 2 Laufwerke 1769,–
PC 1512 mit Farbm. + 1 Laufwerk 1769,–
dto. + 2 Laufwerke 2199,–

Versandkostenpauschale (Warenwert bis DM 1000,–/darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–).
Lieferung nur gegen NN oder Vorauskasse; Ausland nur Vorauskasse.
Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstraße 5, 7324 Rechberghausen, Tel. (07161) 5 28 89

NOCH HÖHERE RECHENGENAUIGKEIT

Obwohl der TI an sich schon für seine hohe Rechengenauigkeit bekannt ist, kann man diese noch weiter erhöhen. Ein kleines, vierzeiliges Programm erhöht diese auf unbegrenzt viele Nachkommastellen. Die Basis dafür ist das schriftliche Teilen, das nun der Computer nach der Eingabe von drei Werten, dem Zähler, dem Nenner und der Anzahl der Stellen, übernimmt.
Durch kleinere Änderungen ist es auch in TI-Basic lauffähig.

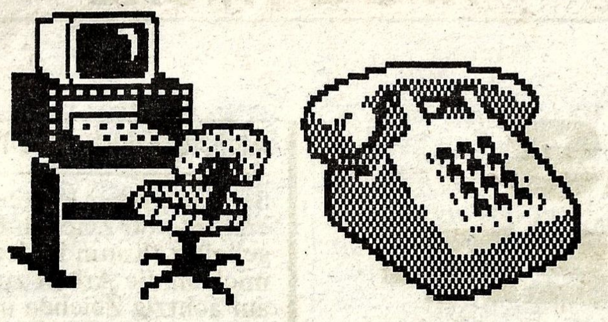
```
10 INPUT "ZÄHLER,
   NENNER, EXPONENT
   ":A,B,E
20 C=A/B :: C=INT(C)
   :: PRINT C
30 D=B*C
40 A=(A–D) *10 E
   :: GOTO 20
```

Michael Brakhage, Kalletal

Vielen Dank für Ihren Tip. Er dürfte vor allem bei den Mathematikern großen Anklang finden.

FORM-FEED BEIM TI-WRITER AUSSCHALTEN

Versuche mit der Formatierung haben bei mir ergeben, daß der automatische Seitenvorschub weitgehend unterdrückt werden kann. In Verbindung mit der TI 99-PP-Schnittstelle von Penta Electronics ist es möglich, wenn man die Ausgabedatei wie folgt angibt:
PRINT.LF.FF=1
Die vollständige Abschaltung, z.B. durch FF=0, ist allerding nicht möglich. Dies ist jedoch, so meine ich, auch nicht unbedingt erforderlich, da dies bereits beim Einspannen des Briefes berücksichtigt werden kann.
Versuche mit einer normalen Centronics konnten leider nicht befriedigen.
Rüdiger Kolbe, Kreuztal



AKTUELL VERLAG MAILBOX
Taeglich 19.00 bis 9.00 Uhr
Wochenende 0.00 bis 24.00 Uhr ONLINE !
Parameter 8 N 1
Tel.: 0 8 9 / 1 8 3 9 5 1

Vielen Dank für Ihren Tip. Sie konnten so zumindest wahrscheinlich den Lesern, die eine TI-99PP-Schnittstelle besitzen, in diesem Problem weiterhelfen.

FEHLER IM BERICHT?

Auf die Anfrage und angebliche Richtigstellung des Herrn Schneider in der Ausgabe 1/87 möchte ich noch einmal den in Heft 11/86 beschriebenen Anschluß des Seikosha GP 100 A erklären. Der 16polige Stecker an der Mechatronic 32k Erweiterung mit Centronics ist gemäß Hersteller wie folgt belegt:
Pin 1 = Strobe
Pin 2–9 = Daten
Pin 10 = Busy
Pin 11 und 16 = Signalerde
Pin 12 = 5 Volt
Im Kundenhandbuch des Druckers findet man u.a. folgende Pinbelegung:
Pin 1–9 = s.o.
Pin 10 = ACK
Pin 11 = Busy
Pin 12 = Gnd
Pin 18 und 36 = 5 Volt
Die Angaben in TI-Revue 11/86 waren also korrekt. Da mein Drucker einwandfrei funktioniert, nehme ich an, daß Herr Schneider eine andere Schnittstelle besitzt. Wichtig bleibt natürlich, daß man die Anschlüsse richtig verbindet.

AKTUELLE CURSORPOSITION

Nach längerem Nachdenken habe ich nun folgende Frage an Sie: Welche Speicherzelle enthält beim TI die momentane Cursorposition?
Rolf Peters, Berkum

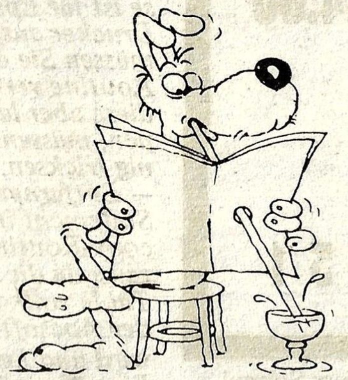
Als Adresse, die die aktuelle Cursorposition enthält, kann man >8320 betrachten. Diese Adresse gibt an, welche Bildschirmausgabe-Startadresse im VDP verwendet wird. Als Nachschlagewerk kann für solche Fragen das Buch 'TI INTERN' von Heiner Martin empfohlen werden.

HARDCOPY VON APESOFT-GRAFIK

Ich habe ein kleines Problem mit meinem TI-99. Ich habe einen TI-99, 32k, P-Box, X-Basic II+ und einen Drucker Seikosha GP-250 X.
Ich arbeite sehr viel mit der Apesoft-Grafik. Nun ist mein Problem, wie bringe ich die Bilder auf meinen Drucker? Ich habe schon alles probiert, es kommt aber lauter Unsinn.
Max Staub-Nanni, CH-Euthal

Um eine Hardcopy der Apesoft-Grafik zu erstellen, können Sie nicht die in der Grafik eingebaute Routine verwenden. Diese ist für Epson-kompatible Drucker ausgelegt. Hierzu müssen Sie eine eigene Routine verwenden. Um diese aber laden zu können, müssen Sie ein wenig tricksen.
– Entfernen Sie das DEF-Statement in der Hardcopy-Routine. Dieses muß raus, da die REF/DEF-Table vollkommen von der Apesoft-Grafik belegt wird und keine weiteren Einträge möglich sind.
– Assemblieren Sie diese so, daß ihre Startadresse auf >B000 liegt.
– Geben Sie in X-Basic CALL APESOFT ein und laden Sie die so präparierte Routine.
– Suchen Sie in der REF/DEF-Table (Adresse >3F00–>3FFF) einen Apesoft-Befehl, den Sie nicht benötigen.
– Überschreiben Sie diesen mit dem Namen Ihrer Hardcopy-Routine (genau 6 Bytes) und der Startadresse (genau 2 Bytes).
Nun können Sie Ihre Routine über CALL LINK aufrufen. Sie müssen jedoch darauf achten, daß Sie keine Teile der Apesoft-Grafik überschreiben, da es sonst zu unkontrollierten Systemabstürzen kommen könnte.

DAS SUPER-SONDER ANGEBOT



Unser Kassetten-Service räumt sein Lager!
Wir bieten an: Bunte Programm-Mischung bestehend aus drei oder fünf Kassetten unterschiedlichen Inhalts mit Spielen, Utilities, Anwender-Programmen. Nur solange Vorrat reicht. Zu Super-Preisen: Das Dreier-Paket für sage und schreibe DM 30, das Fünferpaket für ganze 40 DM! Keine Rücknahme, keine Wunschzusammenstellung! Benutzen Sie unseren Bestell-Coupon im Innern des Heftes, einfach ankreuzen, Geld, Scheck beifügen oder Bankabbuchung. Ist die Ware vergriffen, wird der Scheck nicht eingelöst, beiligendes Bargeld zurückerstattet.

NUTZEN SIE UNSEREN TI-BESTELL-COUPON
AUF SEITE 47

ANPASSUNG AN DIE 80-ZEICHEN-KARTE

Seit Dezember letzten Jahres wird nun endlich die lang ersehnte 80-Zeichen-Karte der Firma Mechatronic ausgeliefert. Kaum in den Rechner eingebaut, machte ich mich an die Arbeit zu versuchen, einige Programme auf achtzig Zeichen umzustellen. Die notwendigen Änderungen für drei Programme, von TI-Forth, Turbo Pasc'99 und dem CorCompManager, habe ich schon herausgefunden.

TI-FORTH Release of Dec. 21,1983
In Screen 54, Zeile 11 müssen die Befehle 'OFF 4 VWTR' zu '7 4 VWTR' umgeändert werden.

TURBO PASC'99 Version 2.0
Hier muß nur die Adresse >3924 von >3B50 nach >3B54 umgeändert werden. Diese Adresse ist auf der Diskette in Sektor >50, beginnend in Byte >24.

CorCompManager Version 2.1 und 2.3
Bei dem Manager sind mehrere Änderungen notwendig. Anhand einer Tabelle werden sie jedoch überschaulicher.

Adresse	Vorher.Wert	Neuer Wert	Sektor	Byte
		Version 2.3		
>D198	>EF16	>EF26	>68	>98
>D1A4	>EF16	>EF26	>68	>A4
>E876	>EF16	>EF26	>7F	>76
>EA92	>EF16	>EF26	>81	>92
>EAA0	>EF16	>EF26	>81	>A0
>D3D2	>EF26	>EF26	>6A	>D2
		Version 2.1		
>D006	>EC30	>EC40	>67	>06
>D012	>EC30	>EC40	>67	>12
>E5C8	>EC30	>EC40	>7C	>C8
>E7E8	>EC30	>EC40	>7E	>E8
>E7F6	>EC30	>EC40	>7E	>F6
>D240	>EC40	>EC40	>69	>40

Seltsamerweise stimmten schon die Adresse >D3D2 bei Version 2.3 und die Adresse >D240 bei Version 2.1.

Martin Trabi

TI ALS WETTER-STATION

Ich möchte meinen TI gerne als Wetterstation einsetzen. Was benötige ich hierzu an Hard- und Software? Können Sie mir weiterhelfen?
Egon Müller,
2905 Edewecht

Um Ihren TI als Wetterstation einsetzen zu können, benötigen Sie zunächst einmal einen A/D-Wandler oder ein anderes I/O-Modul, das Ihnen die Signale, die von Ihrer Station kommen, für den Computer umsetzt und verständlich macht. Eine solche I/O-Erweiterung haben wir schon in verschiedenen TI-Revues zum Selbstbau vorgestellt. Ein Programm, das aber Ihren TI speziell zu einer Wetterstation macht, ist uns nicht bekannt. Vielleicht hat ein TI-Revue-Leser schon in diese Richtung gearbeitet und kann Ihnen weiterhelfen.

EINFÜHRUNG IN GPL: GANZ EINFACH

In dieser Folge beschäftigen wir uns wieder mit ganz einfachen Dingen. Die restlichen Bits des MOVE, die verschiedensten Arten, Werte miteinander zu vergleichen und ein paar neue Befehle sind allesamt nichts Überwältigendes, aber trotzdem nützlich und wichtig. Nur bei den Operanden wird es haarig und das Programm ist diesmal zwar länger als sonst, dafür aber auch ein echtes und spannendes (wer's glaubt wird selig) Spiel.

Der normale RTN setzt das C-Bit auf 0. Viele Subroutinen geben aber über dieses Bit bestimmte Informationen an das Hauptprogramm zurück, so z.B. die Cassettenroutine, ob ein Error vorliegt. Diese benutzen statt RTN die Anweisung RTNC, bei der das C-Bit unverändert bleibt, im Übrigen aber exakt die gleiche Funktion hat.

AUCH FETCH UNTERSTÜTZT SUBROUTINEN

Sie kennen das Verfahren vielleicht schon vom Maschinencode her: Mit BL geht es in eine Subroutine und dort steht dann etwas wie 'MOV *R11+, irgendwas'. Oder 'BLWP' und 'MOV *R14+, irgendwas'. Nach BL bzw. BLWP steht dann ein 'DATA >XXXX', der auf diese Weise gelesen wird, so z.B. die GPL-Adresse nach BLWP &GPLLNK. FETCH macht genau das selbe. Er steht in einer Subroutine, liest das Byte nach CALL, legt es in die durch einen allg. Adr.op. angegebene Adresse und erhöht die Rücksprungadresse auf dem Substack entsprechend. Achtung: FETCH wird dargestellt durch das Byte >88 und >89. Trotzdem ist >89 kein echter DFETCH!

FETCH und DFETCH lesen beide nämlich nur ein Byte, der DFETCH tut aber so, als hätte er zwei Bytes gelesen, von denen das erste ein Nullbyte war.

SCHIEBEBEFEHLE

Mit den Schiebebefehlen ist unsere Liste über arithmetische und logische Befehle komplett:

GPL Befehl	Code (Bin/Hex)	GPL-Statusbyte	Maschinenbefehl
SRA	110111.. (DC–DF)	unbeeinflußt	SRA
SLL	111000.. (E0–E3)	unbeeinflußt	SLA
SRL	111001.. (E4–E7)	unbeeinflußt	SRL
SRC	111010.. (E8–EB)	unbeeinflußt	SRC
GPL	Code (Bin/Hex)	Anzahl der Operanden	

Die Bedeutung der Punkte bei den Maschinenbefehlen der Wert Null als Anzahl der Schiebeschritte eine Verschiebung um 16 Bits.

COMPARE DIES UND COMPARE DAS

Zum Vergleichen von Werten stehen neben den Anweisungen H, GT, CARRY und OVF (siehe Folge IV) noch sieben weitere Statements zur Verfügung. Bevor ich diese im Einzelnen erkläre, zeige ich sie der Übersicht halber erst wieder in einer Tabelle:

GPL-Befehl	Code (Bin/Hex)	Anzahl d. Operanden
CZ	1000111. (8E–8F)	1
CH	110001.. (C4–Cz)	2
CHE	110010.. (C8–CB)	2
CGT	110011.. (CC–CF)	2
CGE	110100.. (D0–D3)	2
CEQ	110101.. (D4–D7)	2
CLOG	110110.. (D8–DB)	2

Sie sehen sofort, daß CZ (Compare if zero) ziemlich aus der Reihe fällt. Er prüft, ob ein Wert gleich Null ist und setzt das C-Bit, wenn dies zutrifft. Alle anderen Anweisungen vergleichen jeweils zwei Werte miteinander und zwar nach diesem Schema:

CH (Compare if high) setzt das C-Bit auf 1, wenn der erste Wert logisch (d.h. absolut) größer dem zweiten ist.

CHE (Compare if high or equal) setzt das C-Bit auf 1, wenn der erste Wert logisch größer oder gleich dem zweiten ist.

CGT (Compare if greater than) setzt das C-Bit auf 1, wenn der erste Wert arithmetisch (d.h., unter Berücksichtigung des ersten Bits als Vorzeichenbit) größer dem zweiten ist.

CGE (Compare if greater or equal) setzt das C-Bit auf 1, wenn der erste Wert arithmetisch größer oder gleich dem zweiten ist.

CEQ (Compare if equal) setzt das C-Bit auf 1, wenn der erste Wert gleich dem zweiten ist.

CLOG (???) setzt das C-Bit auf 0 (!), wenn eine bitweise UND-Verknüpfung zwischen dem ersten und dem zweiten Wert das Ergebnis Null ergibt, d.h., wenn sich nirgendwo zwei Einsen gegenüber stehen.

EINE SACHE MIT FOLGEN: MOVE

In Folge IV konnte ich leider nicht alle Bits des MOVE beschreiben. Dies lag daran, daß MOVE selbst GROMs indiziert adressieren kann, und das Verfahren der indizierten Adressierung hatte ich in dieser Folge erst einmal zeigen müssen. Ist das Bit a nämlich Null und b Eins, so wird die Senke indiziert angesprochen. Ist das Bit d Eins, so wird die Quelle indiziert angesprochen. Beide Male steht dann nach der GROM-Adresse noch ein Byte, welches mit >8300 addiert eine Scratch-Pad-Adr. (Wort!) ergibt, die auch ungerade sein darf und deren Inhalt zu der GROM-Adresse addiert die endgültige Adresse ergibt.

ADRESSOPERANDEN: INDIZIERT INDIREKT

Was soll das? Nun, Sie haben gesehen, daß mit dem Setzen des 4. Bits im Operanden dieser zu einem indirekten Operanden wird. Und Sie wissen, daß das Setzen des 2. Bits ihn zu einem indizierten Operanden macht. Aber was passiert, wenn Sie beide Bits gleichzeitig setzen? Indiziert indirekt – eine seltsame Vorstellung, aber möglich. Sie ist nicht besonders wichtig und meines Wissens auch im gesamten Konsolen-GROM-Bereich nicht verwendet,

aber angeblich kommt diese Konstruktion in Modulen wie z.B. dem Adventure-Modul vor und Sie sollten sie von daher schon kennen.

ALLE CPU-BEREICHE – INDIZIERT INDIREKT

Zunächst wird der Operand wie ein normaler indizierter Operand behandelt und die Adresse dementsprechend errechnet. Danach aber merkt der GPL-Interpreter, daß es sich außerdem noch um einen indirekten Operanden handelt. Er lies aus der Adr. ein Byte, addiert zu diesem >8300 und erhält damit die richtige Adresse.

Beispiel 1:
Der Operand *>8323 <&>8357) liest aus >8357 ein Wort, addiert >8323 hinzu, liest daraus ein Byte, addiert >8300 und erhält die fertige Adresse. Sie beschreiben ihn mit >D023 >57.

Format XI/1:
1101 ++++ ....
* Operand * * Index*

Beispiel 2:
Den Operanden *>1742& >8369) beschreiben Sie mit >DF >9442 >69.

Format XI/2:
1101 1111 ....
....
++++ * Operand * * Index *

ALLE VDP-BEREICHE – INDIZIERT INDIREKT

Beispiel 1:
Der Operand VDP*>0174 &>8317) liest aus >8317 ein Wort, addiert >0174 hinzu und liest daraus das Wort, welches dann die endgültige VDP-Adresse darstellt. Sie schreiben >F174 >17 für ihn.

Format XII/1:
1111 ++++ ....
* Operand * * Index *

Beispiel 2:
Für den Operanden VDP* >2919(&>8353) schreiben Sie >FF >2919 >F3.

Format XII/2:
1111 1111 ....
....
++++ * Operand ** Index*

DAS PROGRAMM

Haben Sie schon einmal einen Intelligenztest gemacht? Nein? Dann haben Sie hier Gelegenheit dazu, vorausgesetzt, Ihnen macht das Abtippen der relativ langen Textliste nichts aus. Das Programm selbst ist nicht schwer zu verstehen, da es alle sich ständig wiederholenden Funktionen in drei Subroutinen vereinigt.

```
G6000  AA00 0000 0000 6010   Header
G6008  0000 0000 0000 0000

G6010  0000 60BB 0F49 4E54   "Intelligenztest"
G6018  454C 4C49 4745 4E5A
G6020  5445 5354
```

Die erste Subroutine schreibt Texte aus einer Liste:

```
G6024 8604   CLR $>8304              Flag
G6026 8909   DFETCH $>8309           Textpointer holen
G6028 D60A   CEQ $>830A,>0A          Sonderformat a ?
      0A
G602B 402F   BR GROM$>602F           Nein: Weiter
G602D 9004   INC $>8304              Ja: Flag
G602F C60A   CH $>830A,>0A           Sonderformat b ?
      0A
G6032 4037   BR GROM$>6037           Nein: Weiter
G6034 A00A   ADD $>830A,$>8301 Ja: korrigieren
      01
G6037 E20A   SLL $>830A,1            Wert verdoppeln
      01
G603A 3300   MOVE >0002 TO $>8305        Tabelle lesen
      0205    FROM GROM$>6181($>8309)
      6181
      09
G6041 8607   CLR $>8307
G6043 3300   MOVE >0001 TO $>8308        Textlänge
      0108    FROM GROM$>0000($>8305)         lesen
      0000
      05
G604A 8E08   CZ $>8308               Länge gleich null ?
G604C 6089   BS GROM$>6089           Ja: Routinenende
G604E 3207   MOVE $>8307 TO VDP$>02E2  Text
      A2E2    FROM GROM$>0001($>8305)     schreiben
      0001
      05
G6055 8E04   CZ $>8304               Sonderformat a ?
G6057 6072   BS GROM$>6072           Nein: Sprung

Sonderformat a: Zähler anzeigen
G6059 BF09   DST $>8309,>02E5  Stadr. Bildschirm
      02E5
G605D AE02   DIV $>8302,>0A          Zähler / 10
      0A
G6060 A203   ADD $>8303,>30          Wert + 48 für ASCII
      30
G6063 9309   DDEC $>8309             Zahl rückw. schreiben
G6065 BCB0   ST VDP*>8309,$>8303
      0903
G6069 BC03   ST $>8303,$>8302  Werte neu ordnen
      02
G606C 8602   CLR $>8302
G606E 8E03   CZ $>8303               Zahl vollständig?
G6070 405D   BR GROM$>605D           Nein: Nächste Stelle
G6072 3502   MOVE >02E0 TO VDP$>0000  Text um eine
      E0A0    FROM VDP$>0020          Zeile scrollen
      00A0
      20
G6079 3500   MOVE >001F TO VDP$>02E1  letzte Zeile
      1FA2    FROM VDP$>02E0          löschen
      E1A2
      E0
G6080 A105   DADD $>8305,$>8307  Pointer
      07                         korrigieren
G6083 9105   DINC $>8305
G6085 8604   CLR $>8304              Flag löschen
G6087 4043   BR GROM$>6043           Auf ein Neues!
G6089 00     RTN
```

Die zweite ist nur eine Zeitverzögerung, die vier verschieden lange Wartezeiten zuläßt:

```
G608A BF05   DST $>8305,>0020        Schleifenwert
      0020                           im Modus 0
G608E 8804   FETCH $>8304            Modus holen
G6090 8A04   CASE $>8304
G6092 40A4   BR GROM$>60A4           Modus 0
G6094 40A0   BR GROM$>60A0           Modus 1
G6096 409C   BR GROM$>609C           Modus 2
G6098 A305   DADD $>8305,>4000       Modus 3
      4000
G609C A305   DADD $>8305,>4000
      4000
G60A0 A305   DADD $>8305,>3200
      3200
G60A4 9305   DDEC $>8305             Schleife
G60A6 40A4   BR GROM$>60A4
G60A8 00     RTN
```

Die letzte scrollt den Bildschirm langsam um eine bestimmte Anzahl von Zeilen nach oben:

```
G60A9 8809   FETCH $>8309            Anz. der Zeilen
G60AB 3502   MOVE >02E0 TO VDP$>0000  Scroll um
      E0A0    FROM VDP$>0020          eine Zeile
      00A0
      20
G60B2 0660   CALL GROM$>608A         kurze Warteschleife
      8A
G60B5 00     DATA >00                Modus 0
G60B6 9209   DEC $>8309              Schleife für Anz.
G60B8 40AB   BR GROM$>60AB           der Zeilen
G60BA 00     RTN
```

Jetzt beginnt das eigentliche Programm:

```
G60BB 0403  BACK >03          grüner Bildschirm
G60BD BEA3  ST VDP$>0380,>10  Farbtabelle für
      8010                              grün
G60C1 3500  MOVE >001F TO VDP$>0381
      1FA3   FROM VDP$>0380
      81A3
      80
G60C8 BF4A  DST $>834A,>0900  Normalen Zeichensatz
      0900                    laden
G60CC 0600  CALL GROM$>0018
      18
G60CF 8700  DCLR $>8300       Zähler nullen
G60D1 0720  ALL >20           Bildschirm löschen
G60D3 0660  CALL GROM$>6024   Titelbild aufbauen
      24
G60D6 01    DATA >01          Text 1
G60D7 0660  CALL GROM$>60A9   um 7 Zeilen scrollen
      A9
G60DA 07    DATA >07
G60DB 0660  CALL GROM$>6024   Text 2
      24
G60DE 02    DATA >02
G60DF 0660  CALL GROM$>60A9   um 2 Zeilen scrollen
      A9
G60E2 02    DATA >02
G60E3 03    SCAN              Warten auf eine
G60E4 40E3  BR GROM$>60E3     Taste
G60E6 0600  CALL GROM$>0034   Piepton
      34
G60E9 0720  ALL >20           Bildschirm löschen
G60EB 0660  CALL GROM$>6024   Text 3
      24
G60EE 03    DATA >03
G60EF 0660  CALL GROM$>60A9   um 6 Zeilen scrollen
      A9
G60F2 06    DATA >06
G60F3 0660  CALL GROM$>608A   Warten (Modus 1)
      8A
G60F6 01    DATA >01
G60F7 0720  ALL >20           Bildschirm löschen
G60F9 0660  CALL GROM$>6024   Text 4
      24
G60FC 04    DATA >04
G60FD 0660  CALL GROM$>608A   Warten (Modus 1)
      8A
G6100 01    DATA >01
G6101 0660  CALL GROM$>60A9   um 3 Zeilen scrollen
      A9
G6104 03    DATA >03
G6105 0660  CALL GROM$>6024   Text 5
      24
G6108 05    DATA >05
G6109 0660  CALL GROM$>60A9   um 1 Zeile scrollen
      A9
G610C 01    DATA >01
G610D 0660  CALL GROM$>6024   Text 6
      24
G6110 06    DATA >06
G6111 0660  CALL GROM$>60A9   um 1 Zeile scrollen
      A9
G6114 01    DATA >01
G6115 03    SCAN              Tastaturabfrage
G6116 D675  CEQ $>8375,>32    Taste "2" ?
      32
G6119 612D  BS GROM$>612D     Ja: Sprung
G611B D675  CEQ $>8375,>31    Taste "1" ?
      31
G611E 4115  BR GROM$>6115     Nein: Neue Abfrage
G6120 0600  CALL GROM$>0034   Piepton bei Taste "1"
      34
G6123 0720  ALL >20           Bildschirm löschen
G6125 0660  CALL GROM$>6024   Text 4
      24
G6128 04    DATA >04
G6129 9100  DINC $>8300       Zähler + 1
G612B 4101  BR GROM$>6101     Routine wiederholen
G612D 0600  CALL GROM$>0034   Piepton bei Taste "2"
      34
G6130 0720  ALL >20           Bildschirm löschen
G6132 0660  CALL GROM$>6024   Text 7
      24
G6135 07    DATA >07
G6136 0660  CALL GROM$>60A9   um 3 Zeilen scrollen
      A9
G6139 03    DATA >03
G613A BD02  DST $>8302,$>8300 Zählerstand kopieren
      00
G613D C700  DCH $>8300,>0006  Zähler größer 6 ?
      0006
G6141 4146  BR GROM$>6146     Nein: Weiter
G6143 BE01  ST $>8301,>07     Ja: Zähler = 7
      07
G6146 0660  CALL GROM$>6024   b: entspr. dem Zähler
      24                      wird ein Text von 11
G6149 0B    DATA >0B          bis 18 angezeigt
G614A 0660  CALL GROM$>60A9   um 5 Zeilen scrollen
      A9
G614D 05    DATA >05
G614E D601  CEQ $>8301,>07    Text 18 (11+7) ?
      07
G6151 4157  BR GROM$>6157     Nein: Weiter
G6153 0660  CALL GROM$>6024   Ja: Text 8 dazu
      24
G6156 08    DATA >08
G6157 0660  CALL GROM$>608A   Warten (Modus 2)
      8A
G615A 02    DATA >02
G615B 8E01  CZ $>8301         Text 11 (11+0) ?
G615D 60CF  BS GROM$>60CF     Ja: Neustart
G615F 0720  ALL >20           Bildschirm löschen
G6161 0660  CALL GROM$>6024   Text 9
      24
G6164 09    DATA >09
G6165 0660  CALL GROM$>60A9   um 4 Zeilen scrollen
      A9
G6168 04    DATA >04
G6169 0660  CALL GROM$>6024   Text 10 (a: Zähler
      24                      wird mit angezeigt)
G616C 0A    DATA >0A
G616D 0660  CALL GROM$>60A9   um 1 Zeile scrollen
      A9
G6170 01    DATA >01
G6171 A601  SUB $>8301,>04  Zähler - 4
      04
G6174 0A    GT              Ergebnis <= 0 ?
G6175 6179  BS GROM$>6179   Nein: Weiter
G6177 8601  CLR $>8301      Ja: Zähler = 0
G6179 0660  CALL GROM$>6024 b: entspr. dem Zähler
      24                    wird ein Text von 19
G617C 13    DATA >13        bis 22 angezeigt
G617D 0660  CALL GROM$>608A Warten (Modus 3)
      8A
G6180 03    DATA >03
G6181 40CF  BR GROM$>60CF   Neustart
```

Startadressen der einzelnen Texte:

```
G6183 61AF 61CE 621B 6234 6264 62A7 62E8 64A1
G6193 64F4 6523 6309 6375 63B8 63C2 63DD 63EF
G61A3 6468 6476 654F 65CB 65CB 6639
```

Und nun das Schönste von allem: Die Texte. Um Programmabstürze durch Tippfehler zu vermeiden, habe ich die zum Lauf wichtigen Bytes unterstrichen. Alles andere ist Text, in dem Tippfehler lediglich Rechtschreibfehler zur Folge hätten.

```
G61AF 1D49 204E 2054 2045 204C 204C 2049 2047 2045
G61C1 204E 205A 2054 2045 2053 2054 0019 5745 4E4E
G61D3 2053 4945 2049 4852 4520 494E 5445 4C4C 4947
G61E5 454E 5A19 5545 4245 5250 5255 4546 454E 2057
G61F7 4F4C 4C45 4E2C 2044 414E 4E17 4452 5545 434B
G6209 454E 2053 4945 2045 494E 4520 5441 5354 4500
G621B 1749 4348 2048 4142 4520 4E55 5220 4549 4E45
G622D 2046 5241 4745 0012 5749 4520 4B41 4E4E 204D
G623F 414E 2044 554D 4D45 1B53 5455 4E44 454E 4C41
G6251 4E47 2042 4553 4348 4145 4654 4947 454E 203F
G6263 001B 5745 4E4E 2053 4945 2044 4945 2041 4E54
G6275 574F 5254 2057 4953 5345 4E19 574F 4C4C 454E
G6287 2C20 4441 4E4E 2044 5255 4543 4B45 4E20 5349
G6299 450B 4449 4520 5441 5354 4520 3100 1957 454E
G62AB 4E20 5349 4520 4441 4E4E 2044 4945 2041 4E54
G62BD 574F 5254 1957 4953 5345 4E2C 2044 414E 4E20
G62CF 4452 5545 434B 454E 2053 4945 0D44 4945 2054
G62E1 4153 5445 2032 0011 4245 5552 5445 494C 554E
G62F3 4720 4948 5245 520D 494E 5445 4C4C 4947 454E
```

```
G6305 5A20 3A00 1753 4945 204B 454E 4E45 4E20 4449
G6317 4553 4553 2053 5049 454C 1C41 4E53 4348 4549
G6329 4E45 4E44 2053 4348 4F4E 2E20 4441 4845 5220
G633B 4953 541A 4553 2053 494E 4E4C 4F53 2C20 4441
G634D 5353 2049 4348 2049 484E 454E 1A45 494E 4520
G635F 4245 5552 5445 494C 554E 4720 5343 4852 4549
G6371 4245 2E00 1353 4945 2053 494E 4420 4549 4E45
G6383 2057 4148 5245 1949 4E54 454C 4C49 4745 4E5A
G6395 4245 5354 4045 2055 4E44 2045 494E 1347 454E
G63A7 4945 2045 5253 5445 4E20 4752 4144 4553 0008
G63B9 5345 4852 2047 5554 0019 5545 4245 5244 5552
G63CB 4348 5343 484E 4954 544C 4943 4820 4755 5400
G63DD 1044 5552 4348 5343 484E 4954 544C 4943 4800
G63EF 1949 4852 2049 4E54 454C 4C49 4745 4E5A 4752
G6401 4144 204C 4945 4754 1C55 4E54 4552 4841 4C42
G6413 2044 4553 2044 5552 4348 5343 484E 4954 5453
G6425 2E19 4441 5320 4953 5420 4142 4553 204E 4943
G6437 4854 2057 4549 5445 5217 5343 484C 494D 4D2C
G6449 2044 4120 4E49 4348 5420 4A45 4445 520E 4549
G645B 4E20 4745 4E49 4520 4953 542E 000C 5345 4852
G646D 204D 4145 5353 4947 001C 4948 5220 494E 5445
G647F 4C4C 4947 454E 5A47 5241 4420 4953 5420 4641
G6491 5354 0C47 4C45 4943 4820 4E55 4C4C 2E00 1C53
G64A3 4945 2053 494E 4420 4445 5348 414C 4220 4549
G64B5 4E45 2047 4546 4148 5217 4655 4552 2044 4945
G64C7 204D 454E 5343 4848 4549 5420 554E 441C 4745
G64D9 484F 4552 454E 2049 4E20 4449 4520 484C 4150
G64ED 534D 5545 404C 452E 0012 4245 4752 5545 4E44
G64FD 554E 4720 4D45 494E 4553 1A55 4E54 4552 5355
G650F 4348 554E 4753 4552 4745 424E 4953 5345 5320
G6521 3A00 1720 2020 202D 204D 414C 2048 4142 454E
G6533 2053 4945 2044 4945 1254 4153 5445 2031 2047
G6545 4544 5255 4543 4B54 2E00 1A44 4153 2048 4549
G6557 5353 542C 2053 4945 2048 4142 454E 2053 4548
G6569 521C 5343 484E 454C 4C20 4449 4553 454E 2044
G657B 554D 4D45 4E20 5343 4845 525A 1B44 5552 4348
G658D 5343 4841 5554 2055 4E44 2044 4945 2054 4153
G659F 5445 2032 1C47 4544 5255 4543 4B54 2E20 4441
G65B1 5320 4953 5420 4549 4E45 2047 5554 4509 4C45
G65C3 4953 5455 4E47 2E00 1C44 4153 2048 4549 5354
G65D5 2C20 5349 4520 4841 4245 4E20 5245 4C41 5449
G65E7 561A 4C41 4E47 4520 4745 4252 4155 4348 542C
G65F9 2055 4D20 4449 4553 4553 1653 5049 454C 205A
G660B 5520 4455 5243 4853 4348 4135 454E 2E1D 4441
G661D 5320 4953 5420 4E49 4348 5420 5345 4852 2052
G662F 5545 484D 4C49 4348 2100 1A44 4153 2048 4549
G6641 5353 542C 2053 4945 2048 4142 454E 2053 4548
G6653 5216 4C41 4E47 4520 4745 4252 4155 4348 542C
G6665 2055 4D20 5A55 1941 4552 4B45 4E2C 2044 4153
G6677 5320 4D41 4E20 5349 4520 4849 4552 1756 4552
G6689 4152 5343 4845 4E20 5749 4C4C 2E20 4441 2049
G669B 5354 1B49 484E 454E 204E 4943 4854 204D 4548
G66AD 5220 5A55 2048 454C 4645 4E2E 00
```

Sven Dyroff

GERÄTESTEUERUNG

In der Industrie werden an Stelle der herkömmlichen Steuerung mit Schützen, Relais und Digitalbausteinen heute mehr und mehr sogenannte „Speicherprogrammierbare Steuerungen" (SPS) eingesetzt. Bei solchen Anlagen ist allein das Programm für die Funktion und Ausführung der Steuerung verantwortlich.

Die Vorteile liegen bei dieser Art auf der Hand: Es muß nun nicht immer eine neue Schaltung aufgebaut oder der gesamte Schaltkreis ausgewechselt werden. Es genügt die Einprogrammierung der Bedingungen und Schaltstellungen.

— Schnelle Anpassung an eine geänderte Aufgabenstellung. Es genügt, die Eingänge und Ausgänge der SPS neu zu programmieren. Man muß keinerlei schaltungstechnische Änderungen vornehmen. Die Änderung erfolgt also nur über die Software.

— Dokumentation der Programmschritte. Wie dieser Punkt schon besagt, handelt es sich nunmehr um ein Programm. Dieses Programm kann, wie jedes Programm, mit Erläuterungen versehen werden. Es liegt also eine Erklärung im Programm als Text vor. Man muß nicht zuerst die Schaltkreise im Schaubild verfolgen.

— Leistungsfähige Diagnose ist möglich. Ein Beispiel ist die Fehlersuche. Bei einem komplexen Schaltkreis steigert diese sich vom langwierigen Suchen ins Unmögliche. — usw. usw. usw...

Hinter einer SPS steckt im Prinzip nichts anderes als ein Computer mit entsprechender Software und den benötigten Ein-/Ausgabe-Kanälen.

Mit dem nun anschließend vorgestellten Programm verhält sich auch der TI-99/4A wie eine solche SPS.

Es ist hiermit möglich, mit dem TI Geräte zu steuern und zu überwachen. Um dies jedoch zu realisieren, benötigt man folgende Grundausstattung:

Neben dem Rechner noch mindestens ein Diskettenlaufwerk, ein bis zwei 8-Kanal I/O-Grundmodule aus TI-REVUE Nr. 2/84, 32k Ram, E/A-Modul und evtl. einen Drucker. Natürlich ist auch noch das Programm „Speicherprogrammierbare Steuerung" notwendig, ohne das die Steuerung überhaupt nicht möglich wäre. Der Drucker ist zwar nicht unbedingt notwendig, aber sehr hilfreich.

Die für die Programmierung einer Steuerung verwendete Sprache ist weitgehend mit der der kommerziellen SPSs identisch. Sie ist in ihrem Aufbau sehr einfach und daher auch leicht zu erlernen.

Bei vollem Ausbau, also mit zwei 8-Kanal I/O-Grundmodulen, stehen dem Programmierer 16 Eingänge, 16 Ausgänge, 64 Merker, das sind Zwischenspeicher für die Signalzustände, 16 Zeitglieder und 16 Zähler zur Verfügung. Auch die Programmierung von Flip-Flops wird unterstützt, so daß selbst komplexere Steuerungen, mit bis zu 3166 Programmschritten, bewältigt werden können.

Es besteht die Möglichkeit, die einzelnen Programmschritte mit Kommentaren zu versehen. Dadurch wird gewährleistet, daß ein Programm übersichtlich bleibt und man es auch noch nach einiger Zeit wieder lesen kann. Es wird auch die Fehlersuche erleichtert, indem man jeden Programmschritt entnehmen kann, was er bewirkt, selbst wenn die Bedeutung des Befehls gerade nicht geläuft ist.

Die Steuerung kann auf Wunsch als Kontaktplan, ähnlich dem Stromlaufplan einer herkömmlichen Schützsteuerung, auf dem Drucker oder Bildschirm ausgegeben werden. Für Testzwecke und zur Fehlersuche steht ein sehr hilfreiches Werkzeug zur Verfügung: Der Testmodus. In ihm können alle Signale, einschließlich der Zeitglieder und Zähler, während die Steuerung läuft, wie mit einem Meßgerät verfolgt werden. Aufwendige Messungen werden somit unnötig.

Die Handhabung des Programms "SPS" ist sehr einfach.

Dadurch, daß die Menü-Technik verwendet wird, sind viele Hilfestellungen schon im Programm gegeben. Die mitgelieferte Anleitung beseitigt ebenfalls noch bestehende Fragen.

Beziehen kann man das Programm „Speicherprogrammierbare Steuerung" bei
Thomas Graf,
Kleeburger Weg 14,
5500 Trier

Der Preis beläuft sich auf 59,— DM. Geliefert wird dann das Programm auf Diskette und ausführlicher Beschreibung.

Für 98,— DM erhält man zusätzlich noch den Quellcode. *Oliver Huber*

DIE QUAL DER WAHL: GEBRAUCHTE PERIBOX ODER NEUES CPS 99-SYSTEM

Für viele User, die jetzt beabsichtigen, sich ein Diskettensystem zuzulegen, stellt sich die Frage, ob sie sich eine gebrauchte P-Box kaufen sollen oder vielleicht das neue CPS 99 System der Firma Atronic.

Der Preis beider Erweiterungssysteme ist ungefähr gleich. Während eine gebrauchte P-Box mit ca. 800,– DM gehandelt wird, ist das CPS-99 mit einem Laufwerk für 999,– DM zu erwerben.

Bei einer gebrauchten P-Box sind in der Regel alle wichtigen Erweiterungskarten enthalten: 32k, RS-232-Karte, Controller und ein Shugart-Laufwerk. Die selbe Ausrüstung erwirbt man auch mit dem CPS-99. Zwischen beiden Systemen bestehen aber zwei gravierende Unterschiede. Zum einen ist der mit dem CPS-System mitgelieferte Diskmanager auf Diskette und nicht, wie bei TI, ein Modul. Zum anderen ist ein Slimline-Laufwerk eingebaut. Dieses Laufwerk verarbeitet double side, double density-Disketten und bietet so eine Speicherkapazität von 360kBytes, während es das in der P-Box enthaltene Laufwerk nur auf 89kBytes bringt. Atronic bietet hier also das Vierfache an Speicherkapazität.

Die P-Box kann man, mit Hilfe eines geeigneten Diskmanagers, auf bis zu vier Laufwerke ausbauen. Für zwei dieser Laufwerke benötigt man dann aber ein getrenntes Gehäuse mit eigener Stromversorgung. Zwei Slimline-Laufwerke haben, wie auch beim CPS-99, im Gehäuse bequem Platz. Das CPS-99-System kann man bis auf drei Laufwerke ausbauen. Beim dritten Laufwerk benötigt man jedoch sowohl ein eigenes Gehäuse mit eigener Stromversorgung als auch ein neues Eprom für das System. Dieses kostet um die 300,– DM. Der Größenunterschied der beiden Erweiterungen ist gewaltig. Während die P-Box einen mehrere Kilo schweren Kasten darstellt, den man, mit Hilfe des Flachbandkabels, an den Rechner anschließt, kann man das sehr viel leichtere CPS-99 direkt an den Rechner anstecken. Durch das Flachbandkabel besteht die Möglichkeit, die P-Box innerhalb eines Spielraums von ca. einem Meter zu plazieren. Das CPS-99 wird ohne Kabel geliefert und hängt somit direkt an der Konsole. Die Laufwerke hat man also direkt „zur Hand". Meiner Meinung nach zu direkt. Für ca. 98,– DM kann man aber von Atronic eine Erweiterung erwerben, die dieses behebt.

In der Geräuschentwicklung zieht die P-Box klar den kürzeren, da sie einen Ventilator eingebaut hat, der die Aufgabe hat, das Netzteil und die Steckkarten zu kühlen. Ein Ausbau ist zwar möglich, stellt sich nur die Frage, wie lange das ungekühlte Netzteil überlebt . . . Sowohl die P-Box als auch das CPS-99 besitzen keinen durchgeschliffenen Bus. Dies bedeutet, daß alle externen Erweiterungen vorher an die Konsole angeschlossen werden müssen, also zwischen Konsole und P-Box bzw. CPS-99.

Ein Riesenvorteil für die P-Box ist das Vorhandensein der insgesamt acht Steckplätze. Einer dieser Steckplätze geht jedoch für das Verbindungskabel Computer-P-Box verloren, dem sogenannten Box-Controller. Es verbleiben also sieben Steckplätze für Erweiterungen wie z.B. dem Disk-Controller, der 32k-Erweiterung etc.

Mit dem CPS-99 können keine Steckkarten verwendet werden. Man kann somit nützliche Erweiterungen wie z.B. die neue Triple-Tech-Karte aus den Staaten, die I/O-Karte und auch die Pascal-Karte abschreiben. Im Falle des Pascals existiert ja noch das Turbo Pascal. Dieses kann aber bei weitem nicht so viel, wie UCSD-Pascal zu bieten hat.

Mit dem CPS-99-System ist mit dem Anfang also auch schon das Ende da. Man kauft sich das CPS-99 und weiter geht nichts mehr. Karten, die hoffentlich in den nächsten Jahren noch erscheinen werden, können nicht verwendet werden. Der CPS-99-User schaut also in die Röhre.

FAZIT

Für einen Preis von ca. 1000,– bietet das CPS-99-System zwar eine handliche Erweiterung mit einem DS/DD-Laufwerk und Controller. Weiterhin ist es leise im Betrieb und kompakt in der Handhabung. Will man seine P-Box auf DS/DD aufrüsten, so kommt das auf einen zusätzlichen Betrag von ca. 800,– DM.

Es besteht jedoch die Möglichkeit einer Aufrüstung durch Steckkarten. Durch dieses Manko an Steckplätzen käme ein CPS-99 für mich nicht in Frage.

Wer der Meinung ist, daß es für den TI in Zukunft keine neuen Steckkarten mehr geben wird, der kann sich getrost ein CPS-99-System zulegen. Für positiv denkende Leute, die sogenannten Optimisten, ist die P-Box angesagt. Sie ist zwar etwas teurer, wenn man 360k Diskettenspeicher erreichen will, aber eben Steckkarten unterzubringen vermag. Übrigens: Bei uns, dem TI-Club Baunatal, gibt es verschiedene Erweiterungen billiger, da wir Rabatte von TI-Firmen erhalten. Diese geben wir gerne an unsere Mitglieder weiter.

TI-Club Baunatal,
Matthias Orf

IM FALLE EINES FALLES

Im Falle eines Falles hilft der berühmte Klebstoff auch nicht mehr, wenn der TI 99/4A nicht mehr will. Selbst wenn wir die Reparatur nicht selbst ausführen können, so ist es doch sinnvoll, erst einmal genau zu ermitteln, wo der Fehler liegt. Der folgende Bericht soll hier einen Leitfaden geben. Dabei wird dann auch auf einige Details eingegangen, die sich im Laufe der Zeit als Schwachstellen beim TI 99/4A herausgestellt haben. Dabei ist der Besitz der Schaltbilder natürlich von sehr hohem Wert. Bei einigen Aussagen, die hier getroffen werden, sind diese sogar zum Verständnis notwendig.

Der TI 99/4A gehört sicher zu den zuverlässigsten Heimcomputern, und, ehrlich gesagt, ist mir bisher noch kein Gerät so ganz von allein kaputt gegangen. Das ist bisher ausschließlich bei fehlerhafter Bedienung (z.B. Herausziehen von P-Box-Karten bei noch eingeschalteter Stromversorgung) oder bei unvorsichtiger Bastelei geschehen. Dennoch, es ist also so weit: Direkt nach dem Einschalten kommt kein Titelbild mehr und der TI 99/4A pfeift nur noch in einem häßlichen Ton, oder aber die Abstürze während des Programlaufes häufen sich. So versuchen wir also den Urheber zu ermitteln. Egal bei welchem Fall, zuerst werden alle Zusatzgeräte von der Konsole abgesteckt und unter allen meine ich hier auch alle, also auch den Kassettenrecorder. Nur der Modulator und der Monitor bzw. Fernseher bleibt dran, wir müssen ja noch etwas sehen, falls überhaupt noch etwas zu sehen ist. Nun schalten wir die Konsole allein ein. Ist der Fehler immer noch vorhanden, so müssen wir hier zwei Fälle unterscheiden: Geht gar nichts mehr, d.h., kommt überhaupt kein Titelbild mehr oder sehen wir wenigstens den Titel noch und können mittels eines Tastendrucks bis zur Auswahlliste kommen.

1. NICHTS GEHT MEHR

Der Fall ist der Schwierigste. Hier können wir eigentlich nur das Netzteil des TI prüfen, es müssen zwei Spannungen, jeweils Wechselspannung, am Stecker zu messen sein. Je nach Ausführung des Trafos liegen diese bei ca. 15 bis 28 V und 10 bis 18 V. Besser wäre es jedoch, wenn wir die Konsole öffnen und die Spannungen direkt nach der Regelung messen. Die Spannungsregelung erfolgt auf der Platine, die sich direkt unter dem Modulschacht befindet. Von dort geht über einen Stecker eine vierpolige Kabelverbindung zur Hauptplatine. An diesem Stecker müssen wir +12 V, +5 V und −5 V messen, jeweils ca. mit einer Toleranz von max. 0,1 V. Wer's kann, sollte die Spannungen auch unter Belastung, sprich, bei eingesteckter Versorgung der Zentralplatine, messen. Fehlt eine der Versorgungsspannungen haben wir den Fehler gefunden, einer der Festspannungsregler (die Typen 7812 und 7905 sind hier eingesetzt) oder der Regeltransistor für +5 V sind dafür die meisten Ursachen. Übrigens, wenn der Regeltransistor getauscht werden muß, brauchen wir nicht die teilweise etwas merkwürdige Bezeichnung des Originals zu suchen (auch TI hat hier verschiedene Typen eingesetzt). Es geht jeder normale pnp-Si-Leistungstransistor mit einer minimalen Strombelastung von 5 A.
Sind alle Versorgungsspannungen vorhanden, so gestaltet sich die Suche schwieriger. Wenn möglich, sollten wir einmal alle auf dem TI 99/4A gestecken Bauelemente(das sind die drei GROMs, der Soundchip und der Videoprozessor) gegen welche aus einer intakten Konsole tauschen. Hilft das auch nicht, sind wir auf Vermutungen angewiesen. Mit aller Vorsicht kann ich hier dann nur zwei Bauelemente nennen, von denen ich schon mehrfach einen Ausfall gehört habe: Das ist einmal der TMS 9901 (der im TI 99/4A als Interrupt- und Tastaturcontroller dient) wie auch der Taktgenerator, der 74LS362.

2. ES KOMMT WAS, ABER NUR MÜLL

Wir schalten also die Konsole ein, auf dem Bildschirm erscheinen aber nur wirre Zeichen. Auch hier müssen wir zwei Fälle unterscheiden. Wir sehen zwar nur wirre Zeichen, aber wenn wir die Tastatur betätigen, funktioniert der TI „wie normal". Daran ist dann sehr wahrscheinlich ein defekter TMS 9929A, also der Videoprozessor, schuld. Da dieser gesteckt ist, können wir von einer intakten Konsole einmal probeweise einen einbauen. Sehen wir aber wirre Zeichen und auch mit der Tastatur rührt sich nichts, dann liegt der Verdacht nicht nur auf dem TMS 9929A, sondern auch bei den RAM's, den TMS 4116. Da sind leider 8 Stück drin und ein Austausch sollte nur komplett vorgenommen werden. Bevor wir aber hier das große Löten anfangen, sollten wir ebenfalls die gesteckten GROM's einmal austauschen. Da Zeichensatz und Titelbild nämlich aus diesen GROMs geladen werden, kann auch hier der Fehler vorliegen.

3. DIE KONSOLE STÜRZT NUR MANCHMAL AB

Nun, hier sollten zu allererst die GROM-Port-Stecker gereinigt werden. Das stand schon mehrfach in der TI-REVUE, hat sich aber wohl noch nicht genug rumgesprochen. 99% dieser nur manchmal auftretenden Abstürze sind auf verschmutzte Kontakte zurückzuführen. Deshalb die Kontakte reinigen, aber bitte mit einem ölfreien Reinigungsmittel wie z.B. Kontakt WL.
Liegt der Fehler nicht in verschmutzten Kontakten, so sollten wir versuchen herauszubekommen, ob der TI immer bei bestimmten Bedingungen abstürzt. Ist dies z.B. immer in einer gewissen Zeit nach dem Einschalten der Fall, so dürfte ein Wärmefehler eines Bauteiles vorliegen. Geübte Elektrobastler können versuchen, diesem Fehler mittels Feuerzeug und Kältespray auf die Schliche zu kommen. Aber bitte, das ist wirklich nur etwas für Geübte.

4. ANDERE FEHLER IN DER KONSOLE

Zwei Defekte, die mir mehr als einmal zu Ohren gekommen sind, sollen hier noch erwähnt werden. Einmal ist das eine defekte Tastatur, d.h., einzelne Tasten gehen nicht. Hier sollte zuerst die Tastatur auf irgendwelche Unterbrechungen untersucht werden. Es gibt Tastaturausführungen, bei denen die Tastanschlüsse abbrechen. Das ist, da hinter der Lötung versteckt, kaum zu sehen. Auch ist das Kabel von der Tastatur zur Hauptplatine bei denjenigen gefährdet, die ihren TI oft auseinandernehmen. Die harten

Kabel brechen gerne ab, bzw. lösen sich von der Verbindung zum Stecker. Also, alles genau durchmessen und wenn dann die Tastatur immer noch nicht funktioniert, dann ist es wohl doch der TMS 9901, der die Tastendrücke so umsetzt, daß es der Prozessor versteht.
Durch Fehlbedienung von außen kann der Kassettenrekorderausgang kaputtgehen. Hier ist fast immer der in der Konsole eingebaute Optokoppler, mit der Bezeichnung TIL 118 defekt. Seltener die Transistoren oder IC-Verstärker im Kassettenrekorderausgang. Das läßt sich aber einfach prüfen: Funktioniert die Aufnahme nicht, ist es der Optokoppler, kommt dagegen gar kein Ton vom Kassettenrekorder, können es die IC-Verstärker sein.

5. KONSOLE ALLEIN FUNKTIONIERT, ABER NICHT MIT DER P-BOX

Der Schaltungsaufwand von Peripheriegeräten ist recht erheblich und so lassen sich auch hier nur ein paar allgemeine Richtlinien geben. Dabei gehen wir natürlich bei der Suche nach dem Schuldigen wieder Schritt für Schritt vor, d.h., bei einzelnen Peripheriegeräten werden diese jeweils nacheinander getestet, die P-Box wird zuerst, ohne daß auch nur eine Karte in ihr steckt, angeschlossen.
Wenn die Konsole nicht mehr will, wenn nur die P-Box angeschlossen ist, so tauschen wir am besten alle in der Karte, an die das Kabel angeschlossen ist, befindlichen ICs aus. Das sind 3 Stück 74LS244 und 1 Stück 74LS245. Im Stecker des Kabels, der in die Konsole eingesteckt wird, befindet sich noch ein 74LS 245. Sehr selten übrigens gehen auch die Treiberbausteine in der Konsole kaputt. Es sind ein 74LS 244 und zwei 74LS367. Sind diese Bausteine in der Konsole defekt, merkt man das übrigens nicht, wenn die Konsole nur allein betrieben wird.
Nun müssen wir defekte Treiberbausteine im Kabel nicht unbedingt schon bemerken, wenn nur die Konsole mit der nackten P-Box betrieben wird. Häufig äußern sich diese auch erst, wenn eine der Karten eingesteckt und auch angesprochen wird. Da die 74LS244 recht bekannt dafür sind, daß sie hie und da einmal ihren Geist aufgeben, so stand es auch schon vor ein paar Jahren in einer amerikanischen Fachzeitschrift, sollten wir diese Bausteine immer im Auge behalten.
Nun kommen wir zu dem Fall, daß sich einzelne Karten nicht mehr ansprechen lassen, d.h., z.B. nach einem OLD DSK1.TEST kommt die lapidare Fehlermeldung I/O ERROR 00 und das natürlich, obwohl der Diskcontroller eingesteckt ist. Mit ziemlich großer Wahrscheinlichkeit liegt das an einem defekten 74LS 259 auf dem Diskcontroller. Diese 74LS259, wie sie auch noch auf der RS232-Karte und diversen Fremdperipheriegeräten eingesetzt werden, sind insbesondere bei Usern, die viel an ihren Geräten basteln, ziemlich gefährdet.
Das gleiche gilt übrigens für die Bausteine mit der Bezeichnung 74LS251, die auch auf dem Disk-Controller und auf der RS232-Karte vorkommen. Ist hier einer defekt, kommt nach dem Einschalten nicht einmal mehr ein Titelbild. Das ist eigentlich

TIPS & TRICKS

auch ein typischer Fehler, wenn die Konsole allein einwandfrei funktioniert und sobald die RS232-Karte oder der Diskcontroller angeschlossen sind, das ganze System nach dem Einschalten nur noch unschön pfeift und der Bildschirm leer bleibt. Bedingt durch die Schaltung des TI 99/4A sind dann meist alle im System befindlichen 74LS251 sowie die TMS 9902 der RS232-Karte „hinüber".
Kommen wir zum letzten Peripheriegerät, welches manchmal für Verwirrung sorgt: die 32K-Speichererweiterung. Liegt hier ein Defekt vor, so denken viele Leute an die Konsole. Das läßt sich aber einfach überprüfen: Wir nehmen ein Extended-Basic-Programm und lassen es einmal nur in der Konsole laufen, d.h., ohne angeschlossene Speichererweiterung. Läuft es hier völlig problemlos, ist die Konsole völlig in Ordnung. Bei angeschlossener, defekter Speichererweiterung kommt dann meist irgendwann Syntax-Error in Line XXXX und wenn wir versuchen, diese Zeile dann zu listen, kommt nur Unfug auf den Bildschirm oder der TI 99/4A hängt sich völlig auf. So etwas liegt dann mit Sicherheit an einem Defekt in der 32K-Speichererweiterung. Um hier das defekte Bauteil zu finden, brauchen wir jemanden, der den Debugger des E/A-Moduls beherrscht (sofern dieser noch läuft, wenn nicht, wird es schwierig). Durch Beschreiben der ganzen Speichererweiterung einmal mit den Werten >FF und dann mit >00, wobei anschließend jeweils die Speichererweiterung auf den korrekten Inhalt geprüft wird, findet man schnell die defekten Speicherstellen. Dann kann das defekte Bauteil anhand des Schaltbildes einfach lokalisiert werden. Wobei hier empfohlen werden soll, immer die 8 Speicherbausteine eines Speicherbereiches komplett zu tauschen. Wenn der Ersatz nämlich in seinen Daten nicht genau den verbleibenden ICs entspricht, kann es zu Funktionsstörungen kommen, wenn man nur einen Baustein wechselt.
Zur genauen Suche der kleineren Defekte gibt es ja auch bei den Userclubs mittlerweile einige Programme, wie sie auch den Clubs noch im letzten Jahr von TI USA zur Verfügung gestellt wurden. Deshalb wurde hier auch nur auf die gravierenden Funktionsstörungen etwas näher eingegangen.
Nun, und wer einen Fehler nicht selber finden kann, braucht auch heute nicht zu verzweifeln, die Fa. Texas Instruments repariert die Geräte immer noch, leider aber ist das dort nicht ganz billig, auch kann man mal bei der Hamburger Fa. Atronic nachfragen. Unabhängig davon sollte man sich jedoch einen Kostenvoranschlag geben lassen, häufig ist es billiger, auf dem Gebrauchtmarkt nach Ersatz zu suchen.
Alle hier gemachten Vorschläge und Fehlerhinweise können natürlich nur unverbindlich sein und sind aus der Erfahrung von nunmehr 6 Jahren TI 99/4A abgeleitet. Eine Gewähr, daß bei einem bestimmten Fehler nur ein ganz bestimmter Baustein defekt ist, kann ich natürlich nicht übernehmen. Dennoch haben sich die erwähnten Bauteile in der Mehrzahl der Fälle als Störenfried erwiesen. Elektronik ist nun einmal so ähnlich wie Lotto, nur umgekehrt. Der 6er entspricht dem Ausfall. Gewisse statistische Häufungen bestimmter Zahlen (Bausteine) kann nicht ausgeschlossen werden!?

Heiner Martin

FORMATIERUNG EINGEBUNDEN

Eine ärgerliche Sache, man befindet sich gerade in einem Dateiverwaltungsprogramm, gibt eifrig neue Adressen oder Posten ein und plötzlich geht nichts mehr. Das Programm verweigert weitere Eingaben, da der Platz auf der Diskette nicht ausreicht. Eine neue Diskette muß her – formatiert natürlich. Doch wer denkt schon immer rechtzeitig an genügend einsatzfähigen Ersatz. Disketten sind da, aber noch frisch verpackt und zum Formatieren muß die Dateiverwaltung verlassen und ein anderes Programm geladen werden.
Je nach Programm und Abspeicherungsart gehen einige bis alle mühselig eingetippten Datensätze verloren. Nachlässigkeit wird unnachgiebig bestraft.

Mit dem folgenden Assemblerprogramm ist damit nun Schluß und der TI 99 drückt noch einmal ein Auge zu. Dieses Programm ist in der Lage, Disketten aus dem Basic zu formatieren. Es kann von einseitig/einfache Dichte bis doppelseitig/doppelte Dichte formatieren. Daneben wird gleichzeitig eine Überprüfung der einzelnen Sektoren vorgenommen, indem jeder Sektor einmal angesprungen wird. Erklärt das Programm alle Sektoren für fehlerfrei, erfolgt in den Sektoren 1 und 2 die Eintragung der Systeminformationen.
Das Programm wurde zum besseren Verständnis ausreichend mit Kommentaren versehen. In der abgedruckten Version läuft es unter XBasic, kann aber auch ohne allzu große Schwierigkeiten für das Editor/Assembler-Modul umgeschrieben werden. Natürlich gehört dazu ein klein wenig Erfahrung im Umgang mit dem TMS-9900-Assembler des TIs.
Ferner ist bei der Umprogrammierung zu beachten, daß die angegebene Scroll-Routine nur vom Extended-Basic zur Verfügung gestellt wird. Für den Editor/Assembler muß diese Routine selbst geschrieben werden.
Und wenn die Programmierkenntnisse schon so weit reichen, dürfte es auch keine Schwierigkeiten mehr bereiten, das Assemblerprogramm in die Menüs bestehender Programme einzubinden, um damit für jeden Notfall gerüstet zu sein.

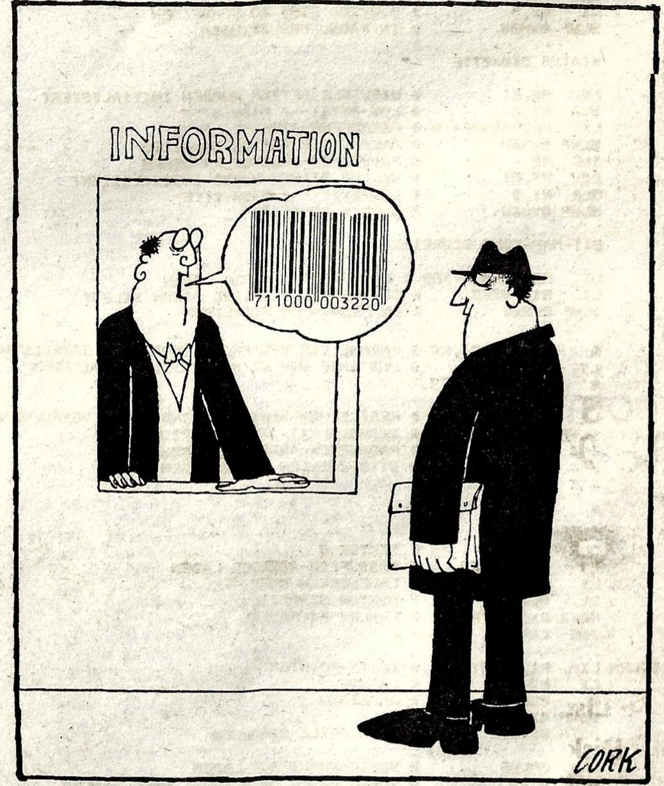
Michael Kovar

```
*  DISKETTENFORMATIERUNG V 1.1
*
*  LAUFFAEHIG UNTER XBASIC & 32K & DISKNR. 1
*
*  AUFRUF MIT: CALL LINK("FORMAT","DISK-NAME",SEITENANZAHL,DICHTE)
*  EINFACHE DICHTE = 1   --   DOPPELTE DICHTE = 2
*  EINSEITIG = 1  --  DOPPELSEIIG = 2
*
       TITL 'DISKETTEN-FORMATIERER FUER BASIC VERSION 1.1'
       IDT  'INIT1.1'
*
       DEF  FORMAT
*
VSBW   EQU  >2020          * UTILITIES
VMBW   EQU  >2024          *    "
STRREF EQU  >2014          *    "
NUMREF EQU  >200C          *    "
XMLLNK EQU  >2018          *    "
ERR    EQU  >2034          *    "

MYWS   BSS  32             * EIGENER WORKSPACE
NAMBUF BYTE 10             * BUFFER FUER DISK-NAME
       BSS  11
DSK    TEXT 'DSK '         * BYTES 0D-10 IN SEKTOR 0
       BYTE >28
       EVEN
TAB    DATA 0,45,90,180    * TABELLE FUER BITMAP IN SEKTOR 0
CFI    EQU  >12B8          * UMRECHNEN FLP IN INTEGER
FAC    EQU  >834A          * FAC-REGISTER
GPLWS  EQU  >83E0          * BASIC-WORKSPACE
SCROLL EQU  >26            * EINE ZEILE SCROLLEN
ERRBV  EQU  >1E00          * ERROR-CODE 'BAD VALUE'
ERRIO  EQU  >2400          * ERROR-CODE 'I/O-ERROR'
PABADR EQU  >1000          * PABBUFFER-ADRESSE
ANZSEK DATA 0,360,720,1440 * TABELLE FUER SEKTORENANZAHL AUF DISKETTE
FORMT1 TEXT '* DISKETTE FORMATIEREN'
FORMT2 TEXT '* SEKTOREN UEBERPRUEFEN'
FORMT3 TEXT '* ALLE SEKTOREN OK'
       EVEN
*
FORMAT LWPI MYWS           * EIGENEN WORKSPACE LADEN
       LI   R5,>2020       * LEERZEICHEN LADEN
       LI   R6,NAMBUF+1    * NAMENSBUFFER LOESCHEN
       LI   R7,10          * 10 BYTES SIND ZU LOESCHEN
LOOP0  MOVB R5,*R6+        * EIN LEERZEICHEN IN BUFFER SCHREIBEN
       DEC  R7             * ZAEHLER ERNIEDRIGEN
       JNE  LOOP0
       LI   R0,>0A00       * MAXIMAL 10 ZEICHEN FUER DISKETTENNAMEN
       MOVB R0,@NAMBUF     * LAENGENBYTE SCHREIBEN
*
SCREEN CLR  R0             * BILDSCHIRM LOESCHEN
       LI   R1,>8000
       LI   R2,768
SCREN1 BLWP @VSBW
       INC  R0
       DEC  R2
       JNE  SCREN1
*
       CLR  R0             * EINFACHE VARIABLE, KEIN ARRAY
       LI   R1,1           * 1. PARAMETER
       LI   R2,NAMBUF      * NAMENSBUFFER (LAENGENBYTE)
       BLWP @STRREF        * STRING HOLEN
       INC  R1             * NAECHSTER PARAMETER
       BLWP @NUMREF        * WIEVIELE SEITEN (1-2)
       BLWP @XMLLNK        * UMRECHNEN IN
       DATA CFI            * INTEGER
       MOV  @FAC,R8        * NACH R8 RETTEN
       JNE  NEXT2          * R8=0 ?
       B    @ERROB         * SPRUNG NACH BASIC MIT ERRORMELDUNG
NEXT2  CI   R8,2           * R8>2 ?
       JLE  NEXT3          * NEIN
       B    @ERROB         * SPRUNG NACH BASIC MIT ERROR
NEXT3  INC  R1             * 3. PARAMETER
       BLWP @NUMREF        * WELCHE DICHTE (EINF./DOPPELT)
       BLWP @XMLLNK        * UMRECHNEN IN
       DATA CFI            * INTERGER
       MOV  @FAC,R9        * NACH R9 RETTEN
       JNE  NEXT4          * R9=0 ?
       B    @ERROB         * RUECKSPRUNG BASIC MIT ERROR
NEXT4  CI   R9,2           * R9>2 ?
       JLE  TEXT1          * NEIN
       B    @ERROB         * RUECKSPRUNG NACH BASIC MIT ERROR
*
TEXT1  LI   R1,FORMT1      * TEXT1-POINTER
       LI   R2,22          * LAENGE 22
       BL   @DSPLAY
       BLWP @XMLLNK        * BETRIEBSSYSTEM-LINK
       DATA SCROLL         * EINE ZEILE SCROLLEN
*
NEXT5  MOV  R9,R7          * R9 KOPIEREN
       SLA  R7,8           * R7 HIGH-BYTE STEHT DIE DICHTE (EINF./DOPPELT)
       A    R8,R7          * R8=R7+R8
       MOV  R7,R5          * DICHTE / SEITENZAHL
       LI   R4,40          * 40 SPUREN
       LI   R3,>0100       * LAUFWERKSNUMMER
       BL   @INIT          * DISK INITIALISIEREN
       MOVB @>8350,R10     * ERROR ABFRAGEN
       JEQ  TEXT2          * KEIN ERROR
       B    @ERROR         * RUECKSPRUNG BASIC MIT ERROR
*
TEXT2  LI   R1,FORMT2      * TEXT2-POINTER
       LI   R2,23          * LAENGE
       BL   @DSPLAY        * ANZEIGEN
       BLWP @XMLLNK
       DATA SCROLL
       BL   @PRUEFE        * SEKTOREN PRUEFEN
*
NEXT1  LI   R0,PABADR      * PABBUFFER-ADRESSE LADEN
       CLR  R1             * R1=CHR$(0)
       LI   R2,256         * 256 BYTES PRO SEKTOR
LOOP1  BLWP @VSBW          * IN PABBUFFER SCHREIBEN
       INC  R0             * PABBUFFER-ADRESSE ERHOEHEN
       DEC  R2             * ZAEHLER ERNIEDRIGEN
       JNE  LOOP1
*
*      SEKTOR 1 SCHREIBEN
*
       LI   R4,1           * SEKTOR 1
       LI   R5,PABADR      * PABBUFFER-ADRESSE LADEN
       LI   R6,>0100       * LAUFWERKSNUMMER
       BL   @WRSEC         * SEKTOR SCHREIBEN
       MOVB @>8350,R10     * ERROR ABFRAGEN
       JNE  ERROR          * RUECKSPRUNG BASIC MIT ERROR
*
*      NAMEN SCHREIBEN
*
NAME   LI   R0,PABADR      * PABBUFFER-ADRESSE LADEN
       LI   R1,NAMBUF+1    * POINTER AUF DISK-NAMEN
       LI   R2,10          * NAME BESTEHT AUS 10 BYTES
       BLWP @VMBW          * IN PABBUFFER SCHREIBEN
*
*      ANZAHL DER SEKTOREN
*
       MOV  R9,R10         * R9 SICHERN
       A    R8,R9          * R9=R8+R9
       DEC  R9             * UM 1 ERNIEDRIGEN
       SLA  R9,1           * MULTIPLIKATION MIT 2
       MOV  R9,R12         * NEUEN WERT SPEICHERN
```

```
       MOV  @ANZSEK(R9),R1 * ANZAHL DER SEKTOREN AUS TABELLE ENTNEHMEN
       LI   RØ,PABADR+1Ø * UND HIGH-BYTE IN PABBUFFER
       BLWP @VSBW        * SCHREIBEN
       INC  RØ           * PABBUFFER-ADRESSE ERHOEHEN
       SWPB R1           * ES FOLGT LOW-BYTE
       BLWP @VSBW        * IN PABBUFFER SCHREIBEN
       MOV  R1Ø,R9       * ALTEN WERT VON R9 HERSTELLEN
*
*      SEKTOREN/SPUR SCHREIBEN
*
       CI   R9,2         * DOPPELTE DICHTE INITIALISIERT ?
       JEQ  NEXT6        * JA --> SPRUNG
       LI   R1,>Ø9ØØ     * 9 SEKTOREN PRO SPUR
       JMP  NEXT7
NEXT6  LI   R1,>12ØØ     * 18 SEKTOREN PRO SPUR
NEXT7  LI   RØ,PABADR+12 * IN PBBUFFER
       BLWP @VSBW        * SCHREIBEN
*
*      DSK SCHREIBEN
*
       LI   RØ,PABADR+13 * PABBUFFER-ADRESSE LADEN
       LI   R1,DSK       * POINTER AUF TEXT
       LI   R2,5         * 5 BYTES SIND ZU SCHREIBEN
       BLWP @VMBW        * IN PABBUFFER BRINGEN
*
*      STATUS DISKETTE
*
       MOV  R8,R1        * WIEVIELE SEITEN WURDEN INITIALISIERT
       SLA  R1,8         * LOW-BYTE --> HIGH-BYTE
       LI   RØ,PABADR+18 * PABBUFFER-ADRESSE LADEN
       BLWP @VSBW        * ANZAHL DER SEITEN SCHREIBEN
       INC  RØ           * PABBUFFER-ADRESSE ERHOEHEN
       MOV  R9,R1        * WELCHE DICHTE WURDE INITIALISIERT
       SLA  R1,8         * LOW-BYTE --> HIGH-BYTE
       BLWP @VSBW        * DICHTE SCHREIBEN
*
*      BIT-MAP-ZONE SCHREIBEN
*
       LI   RØ,PABADR+>38 * PABBUFFER-ADRESSE LADEN
       LI   R1,>Ø3ØØ     * SEKTOREN 1 & 2 SIND SCHON BELEGT
       BLWP @VSBW        * IN PABBUFFER SCHREIBEN
*
       A    @TAB(R12),RØ * ANZAHL DER FREIEN SEKTOREN AUS TABELLE HOLEN
       LI   R2,256       * UND ENDE DER BITMAP-TABELLE ERRECHNEN
       S    @TAB(R12),R2
       AI   R2,-56
       LI   R1,>FFFF     * RESTLICHEN SEKTOREN SIND NICHT VERFUEGBAR
LOOP4  BLWP @VSBW        * DESHALB MIT >FF KENNZEICHNEN
       INC  RØ           * PABBUFFER-ADRESSE ERHOEHEN
       DEC  R2           * BYTE-ZAEHLER ERNIEDRIGEN
       JNE  LOOP4
*
*
*
       CLR  R4           * SEKTOR Ø
       LI   R5,PABADR    * PABBUFFER-ADRESSE LADEN
       LI   R6,>Ø1ØØ     * LAUFWERKSNUMMER
       BL   @WRSEC       * SEKTOR SCHREIBEN
       MOVB @>835Ø,R1Ø   * FEHLER ABFRAGEN
       JNE  ERROR
*
RETURN LI   R1,FORMT3    * TEXT3-POINTER
       LI   R2,18        * LAENGE
       BL   @DSPLAY      * ANZEIGEN
       BLWP @XMLLNK
       DATA SCROLL       * EINE ZEILE SCROLLEN
*
       LWPI GPLWS        * BASIC-WORKSPACE LADEN
       B    @>6A         * RUECKSPRUNG NACH BASIC OHNE FEHLER
*
ERROR  LI   RØ,ERRIO     * ERROR-CODE LADEN
       BLWP @ERR         * RUECKSPRUNG BASIC MIT FEHLER
*
DATA1  DATA Ø            * BUFFER FUER SUBPRG 'PRUEFE'
DATA2  DATA Ø
*
PRUEFE MOV  R11,@DATA2   * RETURNADRESSE SICHERN
       MOV  R9,R2        * R9 KOPIEREN
       A    R8,R2        * ADDIEREN
       DEC  R2           * ERNIEDRIGEN
       SLA  R2,1         * MULTIPLIZIEREN MIT 2
       MOV  @ANZSEK(R2),@DATA1 * SEKTORENZAHL HOLEN
       MOV  @DATA1,R7    * ZAEHLER
       DEC  @DATA1       * ERNIEDRIGEN
       LI   R5,PABADR    * PABBUFFER-ADRESSE
       LI   R6,>Ø1ØØ     * LAUFWERKSNUMMER
LOOP1Ø MOV  @DATA1,R4    * SEKTOR
       BL   @RDSEC       * SEKTOR LESEN
       MOVB @>835Ø,R1    * ERROR ?
       JEQ  LOOP11
       LI   RØ,>22ØØ     * ERROR AUSGEBEN
       BLWP @ERR         * RETURN MIT ERROR
LOOP11 DEC  @DATA1
       DEC  R7
       JNE  LOOP1Ø       * NAECHSTER SEKTOR
       MOV  @DATA2,R11   * RETURNADRESSE HOLEN
       RT
*
*
*
*      SUBPRG SEKTOR LESEN/SCHREIBEN
*
*      FORMAT:           BL   @RDSEC
*                        BL   @WRSEC
*
*      WOBEI             R4-> SEKTORNUMMER
*                        R5-> VDP-BUFFER
*                        R6-> LAUFWERKSNUMMER IM MS-BYTE
*
RDSEC  LI   1,>Ø1ØØ      * FLAG FUER LESEN/SCHREIBEN
       JMP  SEC1
WRSEC  CLR  1            * S. RDSEC
SEC1   MOVB 1,@>834D     * MODUS LESEN ODER SCHREIBEN
       MOV  4,@>835Ø     * SEKTORNUMMER
       MOV  5,@>834E     * PABBUFFER-ADRESSE
       MOVB 6,@>834C     * LAUFWERKSNUMMER
       LI   Ø,>F8Ø       * PAB-ADRESSE
       MOV  Ø,@>8356     * POINTER AUF PAB FUER DSRLNK
       LI   3,>Ø11Ø      * UNTERPROGRAMM >1Ø
       LI   2,2          * 2 BYTES FUER PAB
       LI   1,MYWS+6     * AB R3
       BLWP @VMBW        * PAB SCHREIBEN
       BLWP @DSRLNK      * UNTERPROGRAMM
       DATA 1Ø           * AUFRUFEN
       RT
*
*
*
*
*      UNTERPROGRAMM ZUM INITIALISIEREN EINER DISKETTE
*
*      R3 --> LAUFWERKSNUMMER IM HIGHBYTE
*      R4 --> ANZAHL DER SPUREN PRO SEITE
*      R5 --> HIGHBYTE: DICHTE (>Ø1 EINFACH,>Ø2 DOPPELT)
*             LOWBYTE : SEITENANZAHL
*
*
*      AUFRUF MIT BL  @INIT
*
*
DATENP DATA >Ø111        * PAB-DATEN
INIT   MOVB R3,@>834C    * LAUFWERKSNUMMER
       SLA  R4,8         * HIGHBYTE !!!
       MOVB R4,@>834D    * ANZAHL SPUREN
       MOV  R5,@>835Ø    * SS/SD BIS DS/DD
       LI   R2,>1ØØØ     * PABBUFFER-ADRESSE
       MOV  R2,@>834E    * POINTER AUF PABBUFFER
*
       LI   RØ,>F8Ø      * PABADRESSE
       LI   R1,DATENP    * PABDATEN
       LI   R2,2         * 2 BYTES SCHREIBEN
       BLWP @VMBW        * ADRESSE ERHOEHEN
       MOV  RØ,@>8356    * POINTER FUER DSRLNK SETZEN
       BLWP @DSRLNK      * AUFRUF DES BETRIEBSSYSTEMS
       DATA 1Ø           * ALS UNTERPROGRAMM
       RT                * RUECKSPRUNG ZUM AUFRUFER
*
*
ERROB  LI   RØ,ERRBV     * ERROR-CODE LADEN
       BLWP @ERR         * RUECKSPRUNG BASIC MIT ERROR
*
DSPLAY LI   RØ,738       * ZEILE 24 / SPALTE 3
       MOV  R1,R3        * POINTER KOPIEREN
LOOP8  MOVB *R3+,R1      * EIN ASCII-ZEICHEN HOLEN
       AI   R1,>6ØØØ     * OFFSET >6Ø ADDIEREN
       BLWP @VSBW        * AUF BILDSCHIRM SCHREIBEN
       INC  RØ           * BILDSCHIRMADRESSE ERHOEHEN
       DEC  R2           * ZAEHLER ERNIEDRIGEN
       JNE  LOOP8
       BLWP @XMLLNK
       DATA SCROLL       * EINE ZEILE SCROLLEN
       SETO RØ           * ZAEHLER INITIALISIEREN
       BLWP @GPLLNK      * EINSPRUNG IN GPL
       DATA >34          * ACCEPT-TON
       LIMI 2            * INTERRUPT ERLAUBEN
LOOP9  DEC  RØ           * RØ ERNIEDRIGEN
       JNE  LOOP9        * RØ<>Ø ?
       LIMI Ø            * INTERRUPT UNTERBINDEN
       RT                * RETOUR ZUM AUFRUFER
*
****************************
****************************
**                        **
**  GPLLNK FUER EXT-BASIC **
**                        **
****************************
****************************
*
*
*      FORMAT:           BLWP @GPLLNK
*                        DATA EINSPRUNGADRESSE
*
STKPTR EQU  >8373        * ADRESSE DES STACKPOINTERS
GPLLNK DATA SUBWS,GPLNK1 * EINSPRUNGVEKTOREN
SUBWS  BSS  32           * EIGENER WORKSPACE
GPLNK1 LI   R1,GPLWS+22  * ADRESSE GPL-REGISTER 11
       MOV  *R1+,R2      * REGISTER 11 BIS 15
       MOV  *R1+,R3      * SICHERN
       MOV  *R1+,R4
       MOV  *R1+,R5
       MOV  *R1,R6
       MOV  *R14+,@GPLWS+12 * EINSPRUNGADRESSE NACH R6 DES GPLWS
*
       MOVB @>98Ø2,R7    * MOMENTANE GROM-ADRESSE
       MOVB @>98Ø2,@SUBWS+15  * AUSLESEN
       DEC  R7           * KORRIGIEREN
*
       LWPI GPLWS        * GPLWS LADEN
       LI   R13,>98ØØ    * GROM-WRITE-ADRESSE
       MOVB @STKPTR,R4   * STACKPOINTER HOLEN
       SRL  R4,8         * HIGHBYTE-->LOWBYTE
       AI   R4,>83Ø2     * ADRESSE KOMPLETT
       LI   R5,GPLRTN-2  * RETURNADRESSE LADEN
       MOV  R5,*R4+      * UND AUF STACK LEGEN
       LI   R5,>ØØ3A     * GPL-RETURN ADRESSE KOMMT DARUEBER
       MOV  R5,*R4
       SWPB R4           * LOWBYTE DES STACKPOINTERS
       MOVB R4,@STKPTR   * ABLEGEN
*
       B    @>ØØ6Ø       * GPL-ROUTINE AUSFUEHREN
*
GPLRTN MOVB @STKPTR,R4   * STACKPOINTER HOLEN
       AI   R4,>FEØØ     * ERNIEDRIGEN
       MOVB R4,@STKPTR   * UND ABSPEICHERN
```

```
        LWPI SUBWS          * EIGENER WORKSPACE
        MOVB R7,@>9C02      * GROM-ADRESSE WIEDER
        SWPB R7             * AUF ALTEN WERT
        MOVB R7,@>9C02      * BRINGEN
*
        LI   R1,GPLWS+22    * REGISTER 11 BIS 15
        MOV  R2,*R1+        * DES GPLWS AUF DIE
        MOV  R3,*R1+        * URSPRUENGLICHEN WERTE SETZEN
        MOV  R4,*R1+
        MOV  R5,*R1+
        MOV  R6,*R1
*
        RTWP                * ZURUECK ZUM AUFRUFER
*
************
************
**        **
** DSRLNK **
**        **
************
************
*
*
*       FORMAT              BLWP @DSRLNK
*                           DATA 8
*
*
DSRWS   BSS  >20            * ERLAEUTERUNGEN S. TI-ASSEMBLER SPECIAL
DSRBUF  BSS  20
DSRLNK  DATA DSRWS,DSR0
DSR0    MOV  *R14+,R5
        SZCB @>03B4,R15
        MOV  @>8356,R0
        MOV  R0,R9
        AI   R9,>FFF8
        BLWP @SBR
        MOVB R1,R3
        SRL  R3,8
        SETO R4
        LI   R2,DSRBUF
DSR1    INC  R0
        INC  R4
        CI   R4,>0007
        JGT  IOERR2
        C    R4,R3
        JEQ  DSR2
        BLWP @SBR
        MOVB R1,*R2+
        CB   R1,@>0CF7
        JNE  DSR1
DSR2    MOV  R4,R4
        JEQ  IOERR2
        CLR  @>83D0
        MOV  R4,@>8354
        INC  R4
        A    R4,@>8356
*
DSR3    LWPI >83E0
        CLR  R1
        LI   R12,>0F00
DSR4    MOV  R12,R12
        JEQ  DSR5
        SBZ  0
DSR5    AI   R12,>0100
        CLR  @>83D0
        CI   R12,>2000
        JEQ  IOERR1
        MOV  R12,@>83D0
        SBO  0
        LI   R2,>4000
        CB   *R2,@>000D
        JNE  DSR4
        A    @DSRWS+10,R2
        JMP  DSR7
DSR6    MOV  @>83D2,R2
        SBO  0
DSR7    MOV  *R2,R2
        JEQ  DSR4
        MOV  R2,@>83D2
        INCT R2
        MOV  *R2+,R9
        MOVB @>8355,R5
        JEQ  DSR9
        CB   R5,*R2+
        JNE  DSR6
        SRL  R5,8
        LI   R6,DSRBUF
DSR8    CB   *R6+,*R2+
        JNE  DSR6
        DEC  R5
        JNE  DSR8
DSR9    INC  R1
        BL   *R9
        JMP  DSR6
        SBZ  0
        LWPI DSRWS
        MOV  R9,R0
        BLWP @SBR
        SRL  R1,13
        JNE  IOERR3
        RTWP
IOERR1  LWPI DSRWS
IOERR2  CLR  R1
IOERR3  SWPB R1
        MOVB R1,*R13
        SOCB @>03B4,R15
        AI   R9,>FFFB
        MOV  R9,@>8304
        RTWP
*
*
```

```
SBR     DATA SBRWS,SBR01    * VEKTOREN
*
SBRWS   BSS  32             * WORKSPACE

SBR01   MOV  R13,R8         * ALTEN WORKSPACE KOPIEREN
        MOV  *R8+,R0        * VDP-ADRESSE HOLEN

SBR02   SWPB R0             * LS-BYTE ZUERST
        MOVB R0,@>8C02      * VDP-ADRESSE SETZEN
        SWPB R0             * MS-BYTE
        MOVB R0,@>8C02      * FOLGT
        NOP

SBR03   MOVB @>8800,*R8     * LIEST EIN BYTE AUS VDP-RAM

        RTWP                * RETOUR ZUM AUFRUFER
*
*
        END
```



DANKE!

...für den Kauf von Wohlfahrtsbriefmarken, Ihrem Porto mit Herz & Verstand.

Arbeiterwohlfahrt
Deutscher Caritasverband
Deutscher Paritätischer Wohlfahrtsverband
Deutsches Rotes Kreuz
Diakonisches Werk der EKD
Zentralwohlfahrtsstelle der Juden in Deutschland

B-M-X

Wie gut sind Sie mit einem BMX-Rad? Wollen Sie ein bißchen üben? Na, dann los. Dieses zweiteilige Spiel hat einiges an Schwierigkeitsstufen zu bieten. In mehreren Bildern wird Ihre Geschicklichkeit auf eine harte Probe gestellt.
Das erste Bild beginnt gleich mit einem Doppelsprung über eine Schlucht. Danach ist eine Höhle zu durchqueren, die auch einiges an Gemeinheiten zu bieten hat. Haben Sie die Anfangsschwierigkeiten heil überstanden, können Sie sich an den zweiten Teil des Spieles heranwagen und neue Hindernisse mit Ihrem Geländerad überwinden. Im letzten Bild muß vor dem hohlen Stein eine Kehrtwendung in der Luft erfolgen. Dies erfordert natürlich einen genau berechneten Absprung. Eins sei verraten: Einfacher wird's im zweiten Teil garantiert nicht.
Damit der Spieler nicht so schnell den Mut verliert, stehen ihm pro Durchgang fünf Räder zur Verfügung, die er getrost zu Schrott fahren kann. Nach dem fünften Schrotthaufen muß dann allerdings von vorne begonnen werden.
Das Erreichen des zweiten Teils ist übrigens nicht von der erfolgreichen Durchquerung der ersten Spielstufe abhängig. Er ist ein eigenständiges Programm, das an der Stelle beginnt, an der der erste Teil endet, auf der anderen Seite der Höhle.
Gespielt wird mit dem Joystick (1) bei ausgerasteter Alphalock-Taste. Folgende Funktionen sind möglich:
Joystick nach oben – Sprung
Joystick nach links – anhalten
Joystick nach rechts – weiterfahren
Joystick nach unten – Kopf einziehen
Die letzte Joystickfunktion sollten Sie sich besonders gut einprägen. Bei einem Sprung in der Höhle ist es nicht gerade angenehm, wenn der edelste Körperteil mit einem Stalagtiten in Berührung kommt. Auch die herumfliegenden Steine sorgen manchmal für einen „umwerfenden" Erfolg.
Bei der grafischen Gestaltung der Bildschirme hat sich der Autor schon einige Mühe gegeben, die Trauermusik nach jedem neuen Schrotthaufen strapaziert dabei allerdings die Nerven des Spielers. Am besten den Ton des Fernsehgerätes auf minimale Lautstärke einstellen.

Noch ein Hinweis zum Programm:

Der Autor hat dieses Spiel ganz auf Kassettenbetrieb eingerichtet. Diskettenbesitzer haben es jedoch sofort umgestellt. Jedes RUN "CS1" ist ganz einfach durch RUN "DSK1.Dateiname" zu ersetzen. Damit kann auch am Ende des zweiten Teils wahlweise ganz am Anfang oder gleich mit Teil II neu begonnen werden.

LISTINGS

```
10 ! ***********************
11 ! *                     *
12 ! *    B M X TEIL I     *
13 ! *                     *
14 ! *    COPYRIGHT BY     *
15 ! *                     *
16 ! * DATA-SOFT(AXEL PAPF)*
17 ! *                     *
18 ! *  BENOETIGTE GERAETE *
19 ! *    TI99/4A KONSOLE  *
20 ! *      EXT. BASIC     *
21 ! *    CASS. RECORDER   *
22 ! *      JOYSTICK 1     *
23 ! *                     *
24 ! *  SPEICHERBELEGUNG   *
25 ! *                     *
26 ! *   CA. 9790 BYTES    *
27 ! *                     *
28 ! ***********************
100 W=0 :: H=5 :: CALL CLEAR
 :: CALL SCREEN(2):: FOR I=1
 TO 24 :: PRINT :: NEXT I ::
 CALL COLOR(1,13,13)
110 CALL CHAR(128,"030F03020
787FFFF1F3FFF7F7F1F0E0C0080E
0A0E07071FFFEF0F0E0C0800000"
):: CALL MAGNIFY(3)
120 CALL SPRITE(#1,128,16,60
,30,#2,132,2,60,14)
130 CALL CHAR(132,"0000F0485
17A4646869CE0000000000000000
00020A020203222C0814122")
140 CALL CHAR(136,"0000000C0
60301FFFFFF0001030600C0000000
000000080C0E0E0C08")
145 CALL CHAR(64,"3C4299A1A1
99423C")
150 FOR A=5 TO 9 :: CALL COL
OR(A,16,13):: NEXT A :: FOR
O=3 TO 4 :: CALL COLOR(O,16,
13):: NEXT O
160 CALL MOTION(#2,0,5,#1,0,
5):: FOR I=1 TO 1000 :: NEXT
 I :: CALL MOTION(#2,0,0,#1,
0,0):: FOR I=1 TO 100 :: NEX
T I :: CALL MOTION(#1,18,-20
)
170 FOR I=1 TO 280 :: NEXT I
 :: CALL MOTION(#1,0,0): CA
LL SPRITE(#3,136,2,148,40)::
 FOR I=1 TO 200 :: NEXT I ::
 CALL MOTION(#3,0,10)
175 DISPLAY AT(12,1):"
      @ DEZEMBER 1986"
180 FOR I=1 TO 600 :: NEXT I
 :: CALL MOTION(#3,0,0):: DI
SPLAY AT(20,1):"          B Y
  DATASOFT
                    PRESS SP
ACE TO BEGINN"
185 CALL F
190 CALL DELSPRITE(#1,#3)::
FOR I=1 TO 100 :: NEXT I ::
CALL MAGNIFY(4):: FOR I=1 TO
 100 :: NEXT I :: CALL MAGNI
FY(3)
210 CALL KEY(0,K,S):: IF S=0
 THEN 190 ELSE 220
220 H=5 :: W=0 :: CALL DELSP
RITE(ALL):: CALL CLEAR :: CA
LL SCREEN(2): FOR I=1 TO 24
 :: PRINT :: NEXT I :: CALL
COLOR(1,11,5)
230 CALL CHAR(33,"0C18383030
30381E",35,"0000000800000000
",36,"FFFFFFFFFFFFFFFF")
240 CALL CHAR(37,"0303070F1F
3F7FFF",40,"FFFFFFFFFFF FFFFF
",41,"000000E1FFFFFFFF",42,"
0070F1FFFFFFFFFF":: CALL CO
LOR(2,6,5)
250 CALL CHAR(38,"FFFF766262
20",39,"FEFEFCF1F0F0E0E0")
260 CALL CHAR(132,"000000000
000000000080E04072E53621E00000
00000000000000181414F0305090 8C"
)
270 CALL CHAR(136,"000001010
3071F1F1F1F3F7F7FFFFFFF00008
0E0F0F1F1FCFCFCFEFFFFFFFFFFF"
)
280 CALL CHAR(140,"FF7F1F070
30301010101010101010101FFFEF
EFCF0E0E0C0C0C0808080808")
290 CALL CHAR(64,"0000070F0F
1F1F7F7F3F1F1F0F0301000000084
CEFEFEFEFEFEFCFFFEE0F0F0E")
300 DISPLAY AT(2,1):"  #  #
    #       #     ! #
 #      #      #       #
    #      #       #
  #       #      #"
310 DISPLAY AT(6,1):"#     #
     #     #
 #      #      #       #
           #      #
  #       #      #"
320 CALL CHAR(128,"010100070
E273D090F157BA5A584887080E08
0F018242820E06CA2594141221C"
):: CALL MAGNIFY(3)
330 DISPLAY AT(21,1):"$$$$$$
$$$$$$$$)*)*)%$*)*)%$$$$$$$$$
$$$$$$$$$(((($$(((($$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$"
340 CALL SPRITE(#1,128,2,145
,18)
350 CALL MOTION(#1,0,10)
360 CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1):: IF Y=4
```

```
 THEN GOSUB 38Ø ELSE CALL CO
INC(#1,145,144,2Ø,C):: IF C
THEN 43Ø ELSE 37Ø
37Ø CALL COINC(#1,145,19Ø,1Ø
,C2):: IF C2 THEN 43Ø ELSE C
ALL POSITION(#1,X1,Y1):: IF
Y1>24Ø THEN 46Ø ELSE 36Ø
38Ø CALL MOTION(#1,-4,5):: C
ALL POSITION(#1,X1,Y1):: IF
X1<=1ØØ THEN 39Ø ELSE 38Ø
39Ø CALL MOTION(#1,5,2)::CA
LL POSITION(#1,X1,Y1):: IF X
1>=145 THEN 4ØØ ELSE 39Ø
4ØØ CALL MOTION(#1,Ø,Ø)
41Ø CALL LOCATE(#1,145,Y1)::
 :: CALL COINC(#1,145,144,2Ø
,C):: CALL COINC(#1,145,197,
2Ø,C2):: IF C=-1 OR C2=-1 TH
EN 43Ø ELSE IF C=Ø OR C2=Ø T
HEN 42Ø ELSE 41Ø
42Ø GOTO 35Ø :: RETURN
43Ø CALL MOTION(#1,Ø,Ø,#1,2,
Ø):: CALL POSITION(#1,X1,Y1)
:: IF X1>16Ø THEN 44Ø ELSE 4
3Ø
44Ø CALL MOTION(#1,Ø,Ø):: CA
LL PATTERN(#1,132):: CALL S
:: H=H-1 :: IF H<=Ø THEN 62Ø
 :: GOTO 34Ø
45Ø GOTO 36Ø :: RETURN
46Ø W=1ØØ :: CALL MOTION(#1,
Ø,Ø):: CALL DELSPRITE(ALL)::
 CALL CLEAR :: CALL SCREEN(2
):: FOR I=1 TO 24 :: PRINT :
: NEXT I :: CALL COLOR(1,11,
5)
47Ø DISPLAY AT(2,1):"    #  #
    #        #       ! #
 #       #       #         #
     #       #         #
   #       #       #"
48Ø DISPLAY AT(6,1):"#    #
      #       #
 #       #       #         #
             #       #
   #       #       #"
49Ø DISPLAY AT(11,1):"
$$$$$$$$$$$$$$$$$$$$$$
&&&$$&&&&$$'&&&&&&&&&&"
5ØØ DISPLAY AT(21,1):"$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$"
51Ø CALL SPRITE(#1,128,2,145
,18,#2,136,15,145,1ØØ,#3,14Ø
,11,97,9Ø,#4,64,2,13Ø,25Ø)
52Ø CALL SPRITE(#5,136,16,14
5,116,#6,136,1Ø,145,166,#7,1
4Ø,11,97,14Ø)
53Ø CALL MOTION(#1,Ø,1Ø,#4,Ø
,-1Ø)
54Ø CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1):: CALL C
OINC(ALL,C):: IF C THEN
6ØØ ELSE IF Y=4 THEN GOSUB 5
6Ø ELSE IF X=-4 THEN GOSUB 5
9Ø ELSE IF X=4 THEN 53Ø ELSE
 55Ø
55Ø CALL COINC(ALL,C):: IF C
 THEN GOSUB 6ØØ ELSE CALL PO
SITION(#1,X1,Y1):: IF Y1>24Ø
 THEN 66Ø ELSE 54Ø
56Ø CALL MOTION(#1,-4,4):: C
ALL POSITION(#1,X1,Y1):: CAL
L POSITION(#4,X4,Y4):: CALL
COINC(ALL,Z):: IF Z THEN 6ØØ
 ELSE IF X1<=1ØØ THEN 57Ø EL
SE IF Y1>24Ø THEN 66Ø ELSE 5
6Ø
57Ø CALL MOTION(#1,5,2):: CA
LL POSITION(#1,X1,Y1):: CALL
 COINC(ALL,Z):: IF Z THEN 6Ø
Ø ELSE IF X1>=145 THEN 58Ø E
LSE IF Y1>24Ø THEN 66Ø ELSE
57Ø
58Ø GOTO 53Ø :: RETURN
59Ø CALL MOTION(#1,Ø,Ø):: GO
TO 54Ø :: RETURN
6ØØ CALL LOCATE(#1,145,Y1)::
 CALL MOTION(#1,Ø,Ø):: CALL
PATTERN(#1,132):: CALL S ::
H=H-1 :: IF H<=Ø THEN 61Ø ::
 GOTO 51Ø :: RETURN
61Ø CALL CLEAR :: CALL DELSP
RITE(ALL)
62Ø CALL CHARSET :: CALL DEL
SPRITE(ALL):: CALL CLEAR ::
CALL SCREEN(2):: CALL COLOR(
1,13,13):: FOR I=1 TO 14 ::
CALL COLOR(I,16,13):: NEXT I
63Ø DISPLAY AT(12,1):"LEIDER
 IST DAS SPIEL JETZT
                       ZUENDE
": :"DU HAST";W;"PUNKTE ERRU
NGEN"
64Ø DISPLAY AT(2Ø,1):"NOCH E
IN SPIEL?" :: ACCEPT AT(2Ø,1
7)BEEP VALIDATE("JjNn")SIZE(
1):A$ :: IF A$="N" OR A$="n"
 THEN END
65Ø H=5 :: W=Ø :: GOTO 22Ø :
: RETURN
66Ø CALL MOTION(#1,Ø,Ø):: CA
LL DELSPRITE(ALL):: W=2ØØ ::
 CALL CLEAR :: CALL SCREEN(2
):: FOR I=1 TO 24 :: PRINT :
: NEXT I :: CALL COLOR(1,11,
5)
67Ø CALL CHAR(136,"ØØØØØØ1Ø
1Ø3Ø3Ø7Ø7ØFØF1F1F3F7FFF387CF
```

```
CFCFCFEFEFEFEFEFEFFFFFFFFFFFF"
)
68Ø CALL CHAR(14Ø,"1C3E3F3F3
F3F3F7F7F7F7FFFFFFFFFFFFØØØØØ
Ø8Ø8ØCØCØEØEØFØFØF8F8FCFEFF"
)
69Ø CALL CHAR(68,"FFFFFFFFFF
7F3F3F3F1F1FØFØ7Ø1ØØØØFFFFFF
FFFFFEFCFCF8F8EØCØ8Ø8Ø")
7ØØ DISPLAY AT(2,1):"  #   #
   #       #      ! #
 #       #       #       #
     #       #       #
   #       #       #"
71Ø DISPLAY AT(6,1):"#     #
     #       #
 #       #       #       #
             #       #
   #       #       #"
72Ø DISPLAY AT(11,1):"$$$$$$
$$$$$$$$$$$$$$$$$$$$$$&&&$$&
&&&&&&&&&&&&&&&&&&&& &&&"
73Ø DISPLAY AT(21,1):"$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$"
74Ø CALL SPRITE(#1,128,2,145
,18,#2,136,1Ø,145,4Ø,#3,14Ø,
1Ø,145,165,#4,64,2,13Ø,24Ø)
75Ø CALL SPRITE(#5,68,11,97,
4Ø,#6,64,1Ø,145,1Ø5)
76Ø CALL MOTION(#4,Ø,-1Ø)::
GOTO 78Ø
77Ø CALL MOTION(#1,Ø,1Ø,#4,Ø
,-1Ø)
78Ø CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1):: CALL C
OINC(ALL,Z):: IF Z THEN
84Ø ELSE IF Y=4 THEN GOSUB 8
ØØ ELSE IF X=-4 THEN GOSUB 8
3Ø ELSE IF X=4 THEN 77Ø ELSE
 79Ø
79Ø CALL POSITION(#1,X1,Y1):
: IF Y1>=24Ø THEN 85Ø ELSE C
ALL COINC(ALL,Z):: IF Z THEN
 84Ø ELSE 78Ø
8ØØ CALL MOTION(#1,-5,3):: C
ALL POSITION(#1,X1,Y1):: CAL
L COINC(ALL,Z): IF Z THEN 8
4Ø ELSE IF X1<=1ØØ THEN 81Ø
ELSE IF Y1>24Ø THEN 85Ø ELSE
 8ØØ
81Ø CALL MOTION(#1,5, 3):: CA
LL POSITION(#1,X1,Y1):: CALL
 COINC(ALL,Z):: IF Z THEN 84
Ø ELSE IF X1>=145 THEN 82Ø E
LSE IF Y1>24Ø THEN 85Ø ELSE
81Ø
82Ø GOTO 77Ø :: RETURN
83Ø CALL MOTION(#1,Ø,Ø):: GO
TO 78Ø :: RETURN
84Ø CALL LOCATE(#1,145,Y1)::
 CALL MOTION(#1,Ø,Ø):: CALL
PATTERN(#1,132):: CALL S ::
H=H-1 :: IF H<=Ø THEN 61Ø ::
 GOTO 74Ø :: RETURN
85Ø W=3ØØ :: CALL MOTION(#1,
Ø,Ø):: CALL CLEAR :: CALL DE
LSPRITE(ALL):: CALL SCREEN(2
):: FOR I=1 TO 24 :: PRINT :
: NEXT I :: CALL COLOR(1,11,
5)
86Ø CALL CHAR(136,"ØØØØØ1Ø1Ø
3Ø71F1F1F1F3F7F7FFFFFFFØØØØ8
ØEØFØF1F1FCFCFCFEFFFFFFFFFF"
)
87Ø CALL CHAR(68,"FFFFFFFFFF
7F3F3F3F1F1FØFØ7Ø1ØØØØFFFFFF
FFFFFEFCFCF8F8EØCØ8Ø8Ø")
88Ø DISPLAY AT(2,1):"  #   #
   #       #      ! #
 #       #       #       #
     #       #       #
   #       #       #"
89Ø DISPLAY AT(6,1):"#     #
     #       #
 #       #       #       #
             #       #
   #       #       #"
9ØØ DISPLAY AT(11,1):"$$$$$$
$$$$$$$$$$$$$$$$$$$$$$&&&$$&
&&&&$$&&&&&&&&&&$$$&&&"
91Ø DISPLAY AT(21,1):"$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$ $$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$"
92Ø CALL SPRITE(#1,128,2,145
,18,#2,132,16,145,4Ø,#3,68,1
1,97,39,#4;136,1Ø,145,1Ø1)
93Ø CALL SPRITE(#5,68,11,97,
98,#6,136,1Ø,145,189,#7,68,1
1,97,196)
94Ø GOTO 96Ø
95Ø CALL MOTION(#1,Ø,1Ø)
96Ø CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1):: CALL C
OINC(ALL,Z):: IF Z THEN
1Ø2Ø ELSE IF Y=4 THEN GOSUB
98Ø ELSE IF X=-4 THEN GOSUB
1Ø1Ø ELSE IF X=4 THEN 95Ø EL
SE 97Ø
97Ø CALL POSITION(#1,X1,Y1):
: IF Y1>=24Ø THEN 1Ø3Ø ELSE
CALL COINC(ALL,Z):: IF Z THE
N 1Ø2Ø ELSE 96Ø
98Ø CALL MOTION(#1,-5,3):: C
ALL POSITION(#1,X1,Y1):: CAL
L COINC(ALL,Z):: IF Z THEN 1
Ø2Ø ELSE IF X1<=1ØØ THEN 99Ø
```

```
 ELSE IF Y1>24Ø THEN 1Ø3Ø EL
SE 98Ø
99Ø CALL MOTION(#1,5,3):: CA
LL POSITION(#1,X1,Y1):: CALL
 COINC(ALL,Z):: IF Z THEN 1Ø
2Ø ELSE IF X1>=145 THEN 1ØØØ
 ELSE IF Y1>24Ø THEN 1Ø3Ø EL
SE 99Ø
1ØØØ GOTO 95Ø
1Ø1Ø CALL MOTION(#1,Ø,Ø):: G
OTO 96Ø :: RETURN
1Ø2Ø CALL LOCATE(#1,145,Y1):
: CALL MOTION(#1,Ø,Ø):: CALL
 PATTERN(#1,132):: CALL S ::
 H=H-1 :: IF H<=Ø THEN 61Ø :
: GOTO 92Ø :: RETURN
1Ø3Ø CALL MOTION(#1,Ø,Ø):: C
ALL CHARSET :: CALL CLEAR ::
 CALL DELSPRITE(ALL):: DISPL
AY AT(12,1):"NOCH EINMAL?" :
: ACCEPT AT(12,14)BEEP VALID
ATE("JjNn")SIZE(1):R$ :: IF
R$="Nn" THEN END ELSE 1Ø31
1Ø31 DISPLAY AT(12,1):"NOCH
EINMAL TEIL I": :"ODER WEITE
R MIT TEIL II"
1Ø32 ACCEPT AT(15,1)BEEP VAL
IDATE(DIGIT,"12"):A$ :: IF A
$="2" THEN RUN "CS1" ELSE 22
Ø
1Ø4Ø DATA 75Ø,175,75Ø,175,12
5,175,1ØØØ,175,75Ø,2Ø8,125,1
96,75Ø,196,125,175,75Ø,175,1
25,175,1ØØØ,175,1Ø,4ØØØØ
1Ø5Ø SUB S :: RESTORE 1Ø4Ø :
: FOR C=1 TO 11 :: READ A,B
:: CALL SOUND(-A,B,Ø):: CALL
 SOUND(1,4ØØØØ,Ø):: NEXT C
1Ø6Ø SUBEND
1Ø7Ø SUB F :: RESTORE 1Ø8Ø :
: FOR N=1 TO 34 :: READ T,D
:: CALL SOUND(2Ø*D,T,Ø):: NE
XT N
1Ø8Ø DATA 196,15,131,15,165,
15,165,15,196,15,262,14,262,
14,22Ø,15,196,15,196,15
1Ø9Ø DATA 165,15,131,15,147,
15,165,15,147,15,131,15,147,
15,165,15,147,15,165,15
11ØØ DATA 147,15,165,15,147,
15,131,15,131,15,165,15,196,
15,262,14,247,14,22Ø,15,247,
14,262,14,22Ø,15,196,15
111Ø SUBEND
```

LISTINGS

```
1Ø ! ***********************
11 ! *                     *
12 ! *    B M X TEIL II    *
13 ! *                     *
14 ! *    COPYRIGHT  BY    *
15 ! *                     *
16 ! * DATA-SOFT(AXEL PAPF)*
17 ! *                     *
18 ! * BENOETIGTE GERAETE *
19 ! *   TI99/4A KONSOLE   *
2Ø ! *     EXT. BASIC      *
21 ! *     JOYSTICK 1      *
22 ! *                     *
23 ! *  SPEICHERBELEGUNG   *
24 ! *   CA.13527 BYTES    *
25 ! ***********************
1ØØ CALL CLEAR :: CALL SCREE
N(2):: FOR I=1 TO 24 :: PRIN
T :: NEXT I :: CALL COLOR(1,
13,13)
11Ø CALL CHAR(128,"Ø3ØFØ3Ø2Ø
787FFFF1F3FFF7F7F1FØEØCØØ8ØE
ØAØEØ7Ø71FFFEFØFØEØCØ8ØØØØØ"
):: CALL MAGNIFY(3)
12Ø CALL SPRITE(#1,128,16,6Ø
,3Ø,#2,132,2,6Ø,14)
13Ø CALL CHAR(132,"ØØØØFØ485
17A4646869CEØØØØØØØØØØØØØØØØ
ØØØ2ØAØ2Ø2Ø322CØ814122ØØØØØ"
)
14Ø CALL CHAR(136,"ØØØØØØØCØ
6Ø3Ø1FFFFFFØØØ1Ø3Ø6ØCØØØØØØØ
ØØØØØØØØ8ØCØEØEØCØ8ØØØØØØØØØ"
)
145 CALL CHAR(64,"3C4299A1A1
99423C")
15Ø FOR A=5 TO 9 :: CALL COL
OR(A,16,13):: NEXT A :: FOR
O=3 TO 4 :: CALL COLOR(O,2,1
3):: NEXT O
16Ø CALL MOTION(#2,Ø,5,#1,Ø,
5):: FOR I=1 TO 1ØØØ :: NEXT
 I :: CALL MOTION(#2,Ø,Ø,#1,
Ø,Ø):: FOR I=1 TO 1ØØ :: NEX
T I :: CALL MOTION(#1,18,-2Ø
)
17Ø FOR I=1 TO 28Ø :: NEXT I
 :: CALL MOTION(#1,Ø,Ø):: CA
LL SPRITE(#3,136,2,148,4Ø)::
 FOR I=1 TO 2ØØ :: NEXT I ::
 CALL MOTION(#3,Ø,1Ø)
175 DISPLAY AT(12,1):"
      @ DEZEMBER 1986"
18Ø DISPLAY AT(16,1):"
  T E I L   II"
19Ø FOR I=1 TO 6ØØ :: NEXT I
 :: CALL MOTION(#3,Ø,Ø):: DI
SPLAY AT(2Ø,1):"          BY
  DATASOFT
                  PRESS SP
```

```
ACE TO BEGINN"
195 CALL F
200 CALL DELSPRITE(#1,#3)::
FOR I=1 TO 100 :: NEXT I ::
CALL MAGNIFY(4):: FOR I=1 TO
 100 :: NEXT I :: CALL MAGNI
FY(3)
210 CALL KEY(Ø,K,S):: IF S=Ø
 THEN 200 ELSE 220
220 H=5 :: W=400 :: CALL DEL
SPRITE(ALL):: CALL CLEAR ::
CALL SCREEN(2):: FOR I=1 TO
24 :: PRINT :: NEXT I :: CAL
L COLOR(1,11,5)
230 CALL COLOR(5,7,5):: CALL
 CHAR(64,"FFFFFFFFFFFFFFFF",
65,"ØF1F1F3F7F7F7FFF",66,"Ø3
Ø3Ø7ØFØFØFØFØF")
240 CALL CHAR(67,"CØCØEØEØFØ
FØFØFØ",68,"FØF8F8FCFEFEFEFF
",69,"ØØ18183C3C7E7EFF",7Ø,"
ØØØØØØØØ183C7EFF")
25Ø CALL COLOR(6,13,5):: CAL
L CHAR(72,"FFFFFFFFFFFFFFFF"
)
26Ø CALL CHAR(1ØØ,"ØØØØ1Ø7Ø
E273DØ9ØF157BA5A5848870ØØØØE
ØFØ182428 2ØEØ6CA2594141221C"
)
27Ø CALL CHAR(1Ø4,"ØØØØØØE1F
FFFFFFFFFFFFFFFFFFFFFFFØØ7ØF
8FEFFFFFFFFFFFFFFFFFFFFFFFFF"
)
28Ø CALL CHAR(96,"FFFFFFFFFF
7F3F3F3F1F1FØFØ7Ø1ØØØØFFFFFF
FFFFFEFCFCF8F8EØCØ8Ø8Ø")
29Ø CALL CHAR(33,"ØC1838303Ø
3Ø381E",35,"ØØØØØØØ8ØØØØØØØØ
",36,"FFFFFFFFFFFFFFFF")
3ØØ CALL CHAR(37,"Ø3Ø3Ø7ØF1F
3F7FFF",4Ø,"FFFFFFFFFFFFFFFF
",41,"ØØØØØØE1FFFFFFFFF",42,"
ØØ7ØF1FFFFFFFFF'":: CALL CO
LOR(2,16,5)
31Ø CALL CHAR(38,"FFFF766262
2Ø",39,"FEFEFCF1FØFØEØEØ")
32Ø CALL CHAR(132,"ØØØØØØØØØ
ØØØØØØØ8ØEØ4Ø72E53621EØØØØØ
ØØØØØØØØØØØ181414FØ3Ø5Ø9Ø8C"
)
33Ø CALL CHAR(136,"ØØØØØ1Ø1Ø
3Ø71F1F1F1F3F7F7FFFFFFFØØØØ8
ØEØFØF1F1FCFCFCFEFFFFFFFFFFF"
)
34Ø CALL CHAR(14Ø,"FF7F1FØ7Ø
3 Ø 3 Ø1Ø1Ø1 Ø 1Ø 1Ø1Ø1Ø1Ø1FFFEF
EFCFØEØEØCØCØCØ8Ø8Ø8Ø8ØØØØØ"
)
35Ø CALL CHAR(12Ø,"ØØØØ7ØFØ
F1F1F7F7F3F1F1FØFØ3Ø1ØØØØØØ8
4CEFEFEFEFEFCFFFEEØFØFØEØØØ"
)
36Ø CALL BILD
38Ø DISPLAY AT(11,1):"$$$$$$
$$$$$$$$$$$$$$          &&&$$&
&&&&&&&&&&&&&&&&"
39Ø CALL CHAR(128,"Ø1Ø1ØØØ7Ø
E273DØ9ØF157BA5A584887Ø8ØEØ8
ØFØ182428 2ØEØ6CA2594141221C"
):: CALL MAGNIFY(3)
41Ø DISPLAY AT(21,1):"$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$"
42Ø CALL SPRITE(#1,128,2,145
,18,#2,136,7,145,45,#3,96,11
,97,41,#4,14Ø,11,91,136,#5,1
32,16,145,125)
43Ø CALL SPRITE(#6,14Ø,11,91
,114)
44Ø GOTO 46Ø
45Ø CALL MOTION(#1,Ø,1Ø)
46Ø CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1):: CALL C
OINC(ALL,Z):: IF Z THEN
52Ø ELSE IF Y=4 THEN GOSUB 4
8Ø ELSE IF X=-4 THEN GOSUB 5
1Ø ELSE 47Ø
47Ø IF X=4 THEN 45Ø ELSE CAL
L POSITION(#1,X1,Y1):: IF Y1
>=24Ø THEN 54Ø ELSE CALL COI
NC(ALL,Z):: IF Z THEN 52Ø EL
SE 46Ø
48Ø CALL MOTION(#1,-5,3):: C
ALL POSITION(#1,X1,Y1):: CAL
L POSITION(#4,X4,Y4):: CALL
COINC(ALL,Z):: IF Z THEN 52Ø
 ELSE IF X1<=1ØØ THEN 49Ø EL
SE IF Y1>24Ø THEN 54Ø ELSE 4
8Ø
49Ø CALL MOTION(#1,5,3):: CA
LL POSITION(#1,X1,Y1):: CALL
 COINC(ALL,Z):: IF Z THEN 52
Ø ELSE IF X1>=145 THEN 5ØØ E
LSE IF Y1>24Ø THEN 54Ø ELSE
49Ø
5ØØ GOTO 45Ø :: RETURN
51Ø CALL MOTION(#1,Ø,Ø):: GO
TO 46Ø :: RETURN
52Ø CALL MOTION(#1,Ø,Ø):: CA
LL LOCATE(#1,145,Y1):: CALL
PATTERN(#1,132):: CALL S ::
H=H-1 :: IF H<=Ø THEN 74Ø ::
 GOTO 42Ø
53Ø GOTO 46Ø :: RETURN
54Ø W=5ØØ :: CALL MOTION(#1,
Ø,Ø):: CALL DELSPRITE(ALL)::
 CALL CLEAR :: CALL SCREEN(2
):: FOR I=1 TO 24 :: PRINT :
: NEXT I :: CALL COLOR(1,11,
```

```
5)
550 CALL BILD
570 DISPLAY AT(8,1):"
   AD
   B@@C
   A@@D
  B@@@@C"
580 DISPLAY AT(12,1):"  E
   A@@@@D              B@C
  B@@@@@@@C            A@D
  A@@@@@@@D           B@@@C
 B@@@@@@@@@C          A@@@DF
FFA@@@@@@@@@DFFFFFFFFF"
590 DISPLAY AT(17,1):"HHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHH"
600 DISPLAY AT(21,1):"$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$"
610 CALL SPRITE(#1,128,2,145
,18,#2,120,2,132,245,#3,136,
16,145,40,#4,104,6,157,62,#5
,136,10,145,102)
620 CALL SPRITE(#6,136,10,14
5,200)
630 CALL MOTION(#2,0,-10)::
GOTO 650
640 CALL MOTION(#1,0,10,#2,0
,-10)
650 CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1):: CALL C
OINC(ALL,C):: IF C THEN
720 ELSE IF Y=4 THEN GOSUB 6
80 ELSE IF X=-4 THEN GOSUB 7
10 ELSE IF X=4 THEN 640 ELSE
 660
660 IF Y=-4 THEN GOSUB 670 E
LSE CALL COINC(ALL,C):: IF C
 THEN GOSUB 720 ELSE CALL PO
SITION(#1,X1,Y1):: IF Y1>240
 THEN 780 ELSE 650
670 FOR I=1 TO 10 :: CALL PA
TTERN(#1,100):: NEXT I :: CA
LL PATTERN(#1,128):: GOTO 65
0 :: RETURN
680 CALL MOTION(#1,-5,3):: C
ALL POSITION(#1,X1,Y1):: CAL
L POSITION(#4,X4,Y4):: CALL
COINC(ALL,Z):: IF Z THEN 720
 ELSE IF X1<=100 THEN 690 EL
SE IF Y1>240 THEN 780 ELSE 6
80
690 CALL MOTION(#1,5,3):: CA
LL POSITION(#1,X1,Y1):: CALL
 COINC(ALL,Z):: IF Z THEN 72
0 ELSE IF X1>=145 THEN 700 E
LSE IF Y1>240 THEN 780 ELSE
690
700 GOTO 640 :: RETURN
710 CALL MOTION(#1,0,0):: GO
TO 650 :: RETURN
720 CALL LOCATE(#1,145,Y1)::
 CALL MOTION(#1,0,0):: CALL
PATTERN(#1,132):: CALL S ::
H=H-1 :: IF H<=0 THEN 730 ::
 GOTO 610 :: RETURN
730 CALL CLEAR :: CALL DELSP
RITE(ALL)
740 CALL CHARSET :: CALL DEL
SPRITE(ALL):: CALL CLEAR ::
CALL SCREEN(2):: CALL COLOR(
1,13,13):: FOR I=1 TO 14 ::
CALL COLOR(I,16,13):: NEXT I
750 DISPLAY AT(12,1):"LEIDER
 IST DAS SPIEL JETZT
                       ZUENDE
": :"DU HAST";W;"PUNKTE ERRU
NGEN"
760 DISPLAY AT(20,1):"NOCH E
IN SPIEL?" :: ACCEPT AT(20,1
7)BEEP VALIDATE("JjNn")SIZE(
1):A$ :: IF A$="N" OR A$="n"
 THEN END
770 H=5 :: W=400 :: GOTO 220
 :: RETURN
780 CALL MOTION(#1,0,0):: CA
LL DELSPRITE(ALL):: W=600 ::
 CALL CLEAR :: CALL SCREEN(2
):: FOR I=1 TO 24 :: PRINT :
: NEXT I :: CALL COLOR(1,11,
5)
790 CALL CHAR(136,"000000000
00103070707070F0F1F1F3FFF00000
00000F8FCFEFEFEFFFFFFFFFFFF"
)
800 CALL CHAR(140,"1C3E3F3F3
F3F3F7F7F7F7FFFFFFFFFFFF00000
00080C0C0E0E0F0F0F8F8FCFEFF"
)
810 CALL CHAR(96,"FFFFFFFFF
7F3F3F3F1F1F0F0701000000FFFFF
FFFFFEFCFCF8F8E0C08080")
820 CALL BILD
840 DISPLAY AT(8,1):"
   AD
   B@@C
   A@@D
  B@@@@C"
850 DISPLAY AT(12,1):"  E
   A@@@@D              B@C
  B@@@@@@@C       E    A@D
  A@@@@@@@D  AD  A@D  B@@@C
 B@@@@@@@@@CB@@CB@@@C A@@@DF
FFA@@@@@@@@@DA@@DA@@@DF"
860 DISPLAY AT(17,1):"HHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
```

LISTINGS

```
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHH"
870 DISPLAY AT(21,1):"$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$"
880 CALL SPRITE(#1,128,2,145
,18,#2,120,2,132,9,#3,120,2,
132,220,#4,136,11,145,50,#5,
104,6,157,95)
890 CALL SPRITE(#6,136,11,14
5,156)
900 CALL MOTION(#2,0,-10,#3,
0,-10):: GOTO 920
910 CALL MOTION(#1,0,10,#2,0
,-10,#3,0,-10)
920 CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1):: CALL C
OINC(ALL,Z):: IF Z THEN
990 ELSE IF Y=4 THEN GOSUB 9
50 ELSE IF X=-4 THEN GOSUB 9
80 ELSE IF X=4 THEN 910 ELSE
 930
930 IF Y=-4 THEN GOSUB 940 E
LSE CALL POSITION(#1,X1,Y1):
: IF Y1>=240 THEN 1000 ELSE
CALL COINC(ALL,Z):: IF Z THE
N 990 ELSE 920
940 FOR I=1 TO 10 :: CALL PA
TTERN(#1,100):: NEXT I :: CA
LL PATTERN(#1,128):: CALL CO
INC(ALL,Z):: IF Z THEN 990 :
: GOTO 920 :: RETURN
950 CALL MOTION(#1,-5,3):: C
ALL POSITION(#1,X1,Y1):: CAL
L COINC(ALL,Z):: IF Z THEN 9
90 ELSE IF X1<=100 THEN 960
ELSE IF Y1>240 THEN 1000 ELS
E 950
960 CALL MOTION(#1,5,3):: CA
LL POSITION(#1,X1,Y1):: CALL
 COINC(ALL,Z):: IF Z THEN 99
0 ELSE IF X1>=145 THEN 970 E
LSE IF Y1>240 THEN 1000 ELSE
 960
970 GOTO 910 :: RETURN
980 CALL MOTION(#1,0,0):: GO
TO 920 :: RETURN
990 CALL LOCATE(#1,145,Y1)::
 CALL MOTION(#1,0,0):: CALL
PATTERN(#1,132):: CALL S ::
H=H-1 :: IF H<=0 THEN 730 ::
 GOTO 880 :: RETURN
1000 W=700 :: CALL MOTION(#1
,0,0):: CALL CLEAR :: CALL D
ELSPRITE(ALL):: CALL SCREEN(
2):: FOR I=1 TO 24 :: PRINT
:: NEXT I :: CALL COLOR(1,11
,5)
1010 CALL CHAR(140,"00000000
00000000080E04076E53621E0000
0000000000018141 4F030509 08C
")
1020 CALL BILD
1040 DISPLAY AT(8,1):"
      AD
     B@@C
     A@@D
    B@@@@C"
1050 DISPLAY AT(12,1):"  E
     A@@@@D             B@C
    B@@@@@@C       E    A@D
 E  A@@@@@@D  AD  A@D  B@@@C
B@CB@@@@@@@@CB@@CB@@@C A@@@D
A@DA@@@@@@@@DA@@DA@@@DF"
1060 DISPLAY AT(17,1):"HHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHH"
1070 DISPLAY AT(21,1):"$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$"
1080 CALL SPRITE(#1,128,2,14
5,18,#2,120,2,132,5,#3,120,2
,132,140,#4,136,10,145,60,#5
,136,11,145,105)
1090 CALL SPRITE(#6,104,6,15
7,121,#7,140,16,140,200)
1100 CALL MOTION(#2,0,-9,#3,
0,-10):: GOTO 1120
1110 CALL MOTION(#1,0,10,#2,
0,-9,#3,0,-10)
1120 CALL JOYST(1,X,Y):: CAL
L POSITION(#1,X1,Y1):: CALL
COINC(ALL,Z):: IF Z THEN 119
0 ELSE IF Y=4 THEN GOSUB 115
0 ELSE IF X=-4 THEN GOSUB 11
80 ELSE 1130
1130 IF X=4 THEN 1110 ELSE I
F Y=-4 THEN GOSUB 1140 ELSE
CALL POSITION(#1,X1,Y1):: IF
 Y1>=240 THEN 1200 ELSE CALL
 COINC(ALL,Z):: IF Z THEN 11
90 ELSE 1120
1140 FOR I=1 TO 10 :: CALL P
ATTERN(#1,100):: NEXT I :: C
ALL PATTERN(#1,128):: CALL C
OINC(ALL,Z):: IF Z THEN 1190
 :: GOTO 1120 :: RETURN
1150 CALL MOTION(#1,-8,4)::
CALL POSITION(#1,X1,Y1):: CA
LL COINC(ALL,Z):: IF Z THEN
1190 ELSE IF X1<=100 THEN 11
60 ELSE IF Y1>240 THEN 1200
ELSE 1150
1160 CALL MOTION(#1,8,4):: C
ALL POSITION(#1,X1,Y1):: CAL
```

```
L COINC(ALL,Z):: IF Z THEN 1
19Ø ELSE IF X1>=145 THEN 117
Ø ELSE IF Y1>24Ø THEN 12ØØ E
LSE 116Ø
117Ø GOTO 111Ø
118Ø CALL MOTION(#1,Ø,Ø):: G
OTO 112Ø :: RETURN
119Ø CALL LOCATE(#1,145,Y1):
: CALL MOTION(#1,Ø,Ø):: CALL
 PATTERN(#1,132):: CALL S ::
 H=H-1 :: IF H<=Ø THEN 73Ø :
: GOTO 1Ø8Ø :: RETURN
12ØØ W=8ØØ :: CALL CHARSET :
: CALL MOTION(#1,Ø,Ø):: CALL
 CLEAR :: CALL DELSPRITE(ALL
)
12Ø1 CALL CHAR(128,"Ø1Ø3Ø2Ø6
Ø6Ø1Ø3Ø6Ø6Ø7Ø7Ø3ØØØØØØØ18ØCØ
ØØØØØØ8ØCØ4Ø4ØCØCØ8ØØØØØØ4ØCØ
"):: CALL MAGNIFY(3)
12Ø2 CALL CHAR(132,"ØØØØØ1Ø1
Ø1ØØØØØ1Ø1ØØØØØØØ1Ø1Ø1ØØØØØØ
CØEØCØØØØØ8Ø8ØØØØØØØØØ8Ø8Ø8Ø")
12Ø4 CALL CHAR(14Ø,"ØØØØØØØØ
ØØFØ6Ø7F6F7Ø6ØØØØØØØØØØØØØ7ØF8
181818788E4FØFØFØFØFØF78787C
")
12Ø5 CALL CHAR(136,"ØØØØØE1E
ØE"&RPT$("Ø",47))
12Ø8 CALL SPRITE(#1,128,2,4Ø
,14Ø,#2,132,1Ø,4Ø,14Ø,#3,136
,1Ø,4Ø,174,#4,14Ø,11,4Ø,17Ø)
12Ø9 DISPLAY AT(12,1):"B R A
 V O ": :"DU HAST ES GESCHAF
T": :
121Ø FOR Q=1 TO 14 :: CALL C
OLOR(Q,16,13):: NEXT Q :: DI
SPLAY AT(18,1):"BEI DEINEM W
EG DURCH 8 LEVEL": :"HAST DU
 8ØØ PUNKTE ERGATTERT" :: CA
LL A
1211 CALL DELSPRITE(#3,#4)::
 CALL CHAR(136,"ØØØØØØ783Ø7Ø
BØ7Ø3"&RPT$("Ø",29)):: CALL
SPRITE(#3,136,11,42,151)
122Ø DISPLAY AT(24,1):"NOCH
EINMAL?(J/N)" :: CALL KEY(Ø,
K,S):: IF S=Ø THEN 122Ø ELSE
 IF K=78 OR K=11Ø THEN END E
LSE IF K=74 OR K=1Ø6 THEN 12
3Ø
123Ø DISPLAY AT(24,1):"TEIL
I ODER TEIL II?" :: ACCEPT A
T(24,28)BEEP VALIDATE(DIGIT,
"12")SIZE(-1):Z$ :: IF Z$="1
" THEN 124Ø ELSE IF Z$="2" T
HEN 22Ø
124Ø RUN "CS1"
125Ø DATA 75Ø,175,75Ø,175,12
5,175,1ØØØ,175,75Ø,2Ø8,125,1
96,75Ø,196,125,175,75Ø,175,1
25,175,1ØØØ,175,1Ø,4ØØØØ
126Ø SUB S :: RESTORE 125Ø :
: FOR C=1 TO 11 :: READ A,B
:: CALL SOUND(-A,B,Ø):: CALL
 SOUND(1,4ØØØØ,Ø):: NEXT C
127Ø SUBEND
128Ø SUB A :: FOR Q=1 TO 2 :
: RESTORE 129Ø :: FOR L=1 TO
 16 :: READ T,D :: CALL SOUN
D(D*3Ø,T,Ø):: NEXT L :: NEXT
 Q
129Ø DATA 175,14,262,15,233,
5,22Ø,5,196,5,349,14,262,15,
233,5,22Ø,5,196,5,349,14,262
,15
13ØØ DATA 233,5,22Ø,5,233,5,
196,15
131Ø SUBEND
132Ø SUB F :: RESTORE 133Ø :
: FOR N=1 TO 34 :: READ T,D
:: CALL SOUND(D*15,T,Ø):: NE
XT N
133Ø DATA 131,15,131,15,131,
15,175,15,22Ø,15,131,15,131,
15,131,15,175,15
1331 DATA 22Ø,15,175,15,175,
15,165,15,147,15,147,15,131,
15,131,15,131,15,165,15,196,
15
1332 DATA 131,15,131,15,131,
15,165,15,196,15,262,14,294,
14,262,14,247,14,22Ø,15,196,
15,22Ø,15,175,15,175,15
134Ø SUBEND
1341 SUB BILD
1342 DISPLAY AT(2,1):"  #  #
   #        #      ! #
  #       #       #
#     #     #        #
     #       #       #"
1343 DISPLAY AT(6,1):"#     #
     #        #
  #       #       #
#            #      #
     #       #       #"
1344 SUBEND
```

SATELLITEN-BERECHNUNG

Das Programm berechnet den Seiten- und Höhenwinkel (Azimut und Elevation) einer zu errichtenden terrestrischen Satellitenempfangsantenne für verschiedene vorgegebene Satelliten (auch frei wählbar) für jeden beliebigen Ort (vorgegebene Koordinaten: Kassel).
Es ist ja so, daß jeder Ort andere Richtungswerte für die Antennenaufstellung erfordert.
Die Eingabe der örtlichen Koordinaten und die Ausgabe der Winkelwerte erfolgen in Grad, Minuten und Sekunden.
Die Umrechnung dieser Werte in Dezimalgrad und weiter in Bogenmaß, die Ausrechnung und die Rückumrechnung erfolgen recht schnell und genau.
Ein wichtiger Hinweis: „Der Große ADAC Weltatlas" z.B. führt die Koordinaten aller Orte in folgender Form auf: „Kassel 51.19N 9.30E". Dies bedeutet laut Atlasindex: 51 Grad 19 Minuten nördlicher Breite und 9 Grad 30 Minuten östlicher Länge.
Weiterhin ist zu beachten (im Programm enthalten!), daß südliche Breitengrade und westliche Längengrade mit negativem Vorzeichen einzugeben sind.
Die Führung durch das Programm erfolgt logisch Schritt für Schritt.
Guten Erfolg bei der Ausrichtung des Antennenspiegels!

VERTEILUNGSRECHNUNG

Das Programm „Verteilungsrechnung" hilft Ihnen bei der Verteilung von Gegenständen oder Geldwerten, auch kann die Gewinnverteilung einer Personengesellschaft (Einzelunternehmen, OHG) berechnet werden. Unter Punkt 1 des Menüs verbirgt sich die normale Verteilungsrechnung, mit diesem Programm können Sie private Verteilungsprobleme lösen. Außer der Verteilungssumme, den Anteilen der 1 – 10 Personen, werden auch Vor- und Nachleistungen berücksichtigt. Vorleistungen hat eine Person schon vor der Gewinnausschüttung erhalten, sie sind nicht mehr im Gesamtbetrag enthalten. Nachleistungen kann eine einzelne Person für besondere Arbeiten bzw. Leistungen erhalten, diese sind in der Gesamtsumme noch enthalten.
Punkt 2 dient der Berechnung von Gewinnausschüttungen bei Personengesellschaften, hier wird außerdem noch eine Verzinsung des Stammkapitals der einzelnen Gesellschafter vorgenommen, diesen Zinssatz können Sie vorgeben (4% beträgt der in der Wirtschaft übliche Zinssatz), außerdem wird der Rest der Verteilungssumme, Gewinnausschüttung nach Abzug der Vor- und Nachleistungen und der Verzinsung, nicht nach Anteilen verteilt, sondern nach Köpfen.
Die Ausgabe der Gewinnanteile erfolgt 8 Stellen vor und 2 Stellen nach dem Komma, so daß das Programm leicht zur Berechnung der Gewinnausschüttung eines mittelgroßen Betriebes ausreicht.

```
10 !***********************
12 !*                     *
14 !* Satellitenempfangs- *
16 !* Antennen:Berechnung *
18 !*    von Azimut und   *
20 !*      Elevation      *
22 !*                     *
24 !*                     *
26 !*                     *
28 !*    Copyright by     *
30 !*                     *
32 !*  Gottfried Aresin   *
34 !*                     *
36 !*                     *
38 !* Benoetigte Geraete: *
40 !*  TI-99/4A Konsole   *
42 !*     Ext. Basic      *
44 !*                     *
46 !*                     *
48 !*                     *
50 !*  Speicherbelegung   *
52 !*                     *
54 !*     6556 Bytes      *
56 !*                     *
58 !*                     *
60 !***********************
64 !  Unterlagen:
66 !1. ELEKTOR Nr. 190,
68 !  Oktober 1986, S. 24
70 !2. Funkschau 17/86,
72 !  Poster seite 51-54
74 !3. Prospekt der Firma
76 !  ASTRO
78 !4. "Der Grosse ADAC-
80 !  Weltatlas"
82 !
84 !
100 CALL CLEAR
110 CALL SCREEN(5)
120 FOR I=1 TO 12
130 CALL COLOR(I,16,5)
140 NEXT I
150 CALL COLOR(0,10,5)
160 CALL CHAR(64,"00000000FF
000000")
170 PRINT
180 CALL CHAR(91,"0044003844
7C4444")!AE
190 CALL CHAR(92,"0044003844
444438")!OE
200 CALL CHAR(93,"0044004444
444438")!UE
210 CALL CHAR(123,"000044003
8447C44")!ae
220 CALL CHAR(124,"000044003
8444438")!oe
230 CALL CHAR(125,"000044004
4444438")!ue
240 CALL CHAR(126,"000070485
0484448")!sz
```

LISTINGS

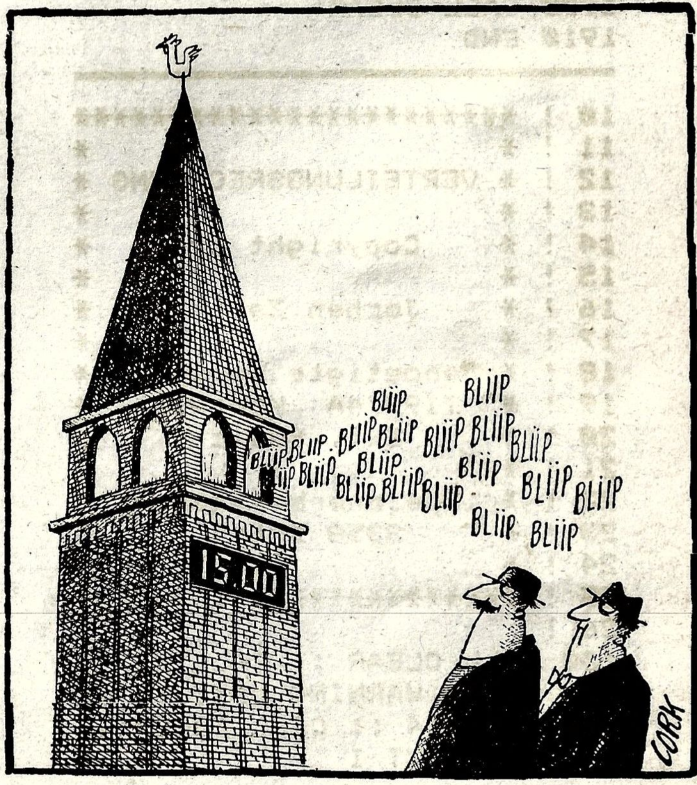
```
250 PRINT "* Satellitenempfa
ngsspiegel*"
260 PRINT "* @@@@@@@@@@@@@@@@
@@@@@@@@@@@*"
270 PRINT "          Berechnung
 von"
280 PRINT "   Seiten- und H!
henwinkel"
290 PRINT "    (Azimuth und E
levation)"
300 PRINT "*   @@@@@@@@@@@@@@@
@@@@@@@@@ *"
310 PRINT
320 PRINT
330 PRINT " Der Satellit bef
indet sich"
340 PRINT " geostation(r in
36.000 km "
350 PRINT "     H!he )ber dem
[quator;"
360 PRINT " dies entspricht
etwa dem "
370 PRINT "     6,5-fachen Er
dradius."
380 PRINT "*@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*"
390 PRINT
400 PRINT "               Ta
ste dr)cken"
410 PRINT : : :
420 CALL KEY(0,K,S):: IF S=0
 THEN 420
430 PRINT
440 CALL CHAR(36,"0030484830
")!GRAD
450 CALL CHAR(37,"0020202040
")!MINUTEN
460 CALL CHAR(38,"0024242448
")!SEKUNDEN
470 CALL CLEAR
480 PRINT "  Satellitenposi
tionen:"
490 PRINT
500 PRINT "@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@"
510 PRINT
520 PRINT "0 G
  31$        W"
530 PRINT "1 INTELSAT F 11
  27$ 30% W"
540 PRINT "2 TV-SAT
  19$        W"
550 PRINT "3 SOWJETBL.
   1$        W"
560 PRINT "4 TELE-X
   5$        O"
570 PRINT "5 ECS 2
   7$        O"
580 PRINT "6 ECS 1 (EUTELSAT
) 13$        O"
590 PRINT "7 SES (LUX)
  19$        O"
600 PRINT "8 DFS(Kopernikus)
  23$ 30% O"
610 PRINT "9 INTELSAT F 12
  60$        O"
620 PRINT "* Freie Berechnun
g:"
630 PRINT
640 PRINT "@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@"
650 PRINT
660 PRINT "Auf welchen Satel
liten soll": :"  die Antenne
 ausgerichtet": :"       werde
n? (0-9)"
670 DISPLAY AT(23,20):"6"
680 ACCEPT AT(23,20)SIZE(-1)
BEEP VALIDATE(DIGIT,"*"):A$
690 IF A$="0" THEN B$="G" ::
 C$="-031" :: D$="00"
700 IF A$="1" THEN B$="F11"
:: C$="-027" :: D$="30"
710 IF A$="2" THEN B$="TV-SA
T" :: C$="-019" :: D$="00"
720 IF A$="3" THEN B$="SOWJE "
TBL." :: C$="-001" :: D$="00
730 IF A$="4" THEN B$="TELE-
X" :: C$="+005" :: D$="00"
740 IF A$="5" THEN B$="ECS 2
" :: C$="+007" :: D$="00"
750 IF A$="6" THEN B$="ECS 1
(EUTELSAT)" :: C$="+013" ::
D$="00"
760 IF A$="7" THEN B$="SES (
LUX)" :: C$="+019" :: D$="00
"
770 IF A$="8" THEN B$="DFS(K
opernikus)" :: C$="+023" ::
D$="30"
780 IF A$="9" THEN B$="INTEL
SAT F12" :: C$="+060" :: D$=
"00"
790 IF A$="*" THEN B$="Freie
 Berechnung:" :: C$="+000" :
: D$="00"
800 CALL CLEAR
810 DISPLAY AT(7,1):"Eingabe
 des L(ngengrades des"
820 DISPLAY AT(9,1):"   Sate
lliten: +000$,00%,00&"
830 DISPLAY AT(9,16)SIZE(4):
C$
840 DISPLAY AT(9,22)SIZE(2):
D$
850 DISPLAY AT(10,1):"
         @@@@@@@@@@@@@"
860 DISPLAY AT(12,1):B$
870 DISPLAY AT(17,1):"WICHTI
G! Westlich von"
```

```
880 DISPLAY AT(18,1):"Greenw
ich '-' Vorzeichen!"
890 ACCEPT AT(9,16)SIZE(-4)B
EEP VALIDATE(NUMERIC," "):A
900 IF A>180 OR A<-180 THEN
890
910 ACCEPT AT(9,22)SIZE(-2)B
EEP VALIDATE(DIGIT," "):B
920 IF B>=60 THEN 910
930 ACCEPT AT(9,26)SIZE(-2)B
EEP VALIDATE(DIGIT," "):C
940 IF C>=60 THEN 930
950 !Umrechnung in Dezimalgr
ad
960 D=B/60
970 E=C/3600
980 IF A<0 THEN F=A-D-E :: G
OTO 1000
990 F=A+D+E
1000 IF F>180 OR F<-180 THEN
820
1010 DISPLAY AT(24,1)BEEP:"$
(DEZ)=";F
1020 CALL CLEAR
1030 DISPLAY AT(1,1):"  Ante
nnenaufstellungsort"
1040 DISPLAY AT(2,1):"* @@@@
@@@@@@@@@@@@@@@@@@@@  *"
1050 DISPLAY AT(4,1):"Geogra
phischer Breitengrad:"
1060 DISPLAY AT(5,1):"
  +51$,19%,00&"
1070 DISPLAY AT(6,1):"
  @@@@@@@@@@@@"
1080 DISPLAY AT(8,1):"Vorgeg
ebener Ort: KASSEL"
1090 DISPLAY AT(10,1):"WICHT
IG! Westlich von"
1100 DISPLAY AT(11,1):"Green
wich '-' Vorzeichen"
1110 ACCEPT AT(5,9)SIZE(-3)B
EEP VALIDATE(NUMERIC," "):H
1120 IF H>90 OR H<-90 THEN 1
110
1130 ACCEPT AT(5,14)SIZE(-2)
BEEP VALIDATE(DIGIT," "):I
1140 IF I>=60 THEN 1130
1150 ACCEPT AT(5,18)SIZE(-2)
BEEP VALIDATE(DIGIT," "):J
1160 IF J>=60 THEN 1150
1170 DISPLAY AT(8,1):""
1180 !Umrechnung in Dezimalg
rad
1190 K=I/60
1200 L=J/3600
1210 IF H<0 THEN M=H-K-L ::
GOTO 1230
1220 M=H+K+L
1230 IF M>90 OR M<-90 THEN 1
060
1240 'DISPLAY AT(12,1)BEEP:"
$(Dez)=";M
1250 DISPLAY AT(14,1):"Geogr
aphischer L{ngengrad"
1260 DISPLAY AT(15,1):"
  +009$,30%,00&"
1270 DISPLAY AT(16,1):"
  @@@@@@@@@@@@@"
1280 DISPLAY AT(18,1):"Vorge
gebener Ort: KASSEL"
1290 DISPLAY AT(20,1):"WICHT
IG! S}dlich des"
1300 DISPLAY AT(21,1):"[quat
ors '-' Vorzeichen!"
1310 ACCEPT AT(15,8)SIZE(-4)
BEEP VALIDATE(NUMERIC," "):N
1320 IF N>180 OR N<-180 THEN
 1310
1330 ACCEPT AT(15,14)SIZE(-2
)BEEP VALIDATE(DIGIT," "):O
1340 IF O>=60 THEN 1330
1350 ACCEPT AT(15,18)SIZE(-2
)BEEP VALIDATE(DIGIT," "):P
1360 IF P>60 THEN 1350
1370 DISPLAY AT(18,1):""
1380 !Umrechnung in Dezimalg
rad
1390 Q=O/60
1400 R=P/3600
1410 IF N<0 THEN S=N-Q-R ::
GOTO 1430
1420 S=N+Q+R
1430 IF S>180 OR S<-180 THEN
 1260
1440 !DISPLAY AT(22,1)BEEP:"
$(Dez)=";S
1450 !Umrechnung in Bogenmas
s
1460 T=F*PI/180 !Satellit
1470 U=M*PI/180 !Breitengrad
1480 V=S*PI/180 !Laengengrad
1490 !Berechnung Azimuth
1500 AZ=ATN(TAN(V-T)/SIN(U))
1510 AZI=180+AZ*180/PI
1520 !Berechnung Elevation
1530 EL=ATN((COS(U)*COS(V-T)
-.1513)/SQR(1-(COS(U)*COS(V-
T))^2))
1540 ELE=EL*180/PI
1550 !DISPLAY AT(23,1):"Azim
uth=";AZI;"$"
1560 !DISPLAY AT(24,1):"Elev
ation=";ELE;"$"
1570 A1=INT(AZI)
1580 A2=AZI-A1
1590 A3=A2*60
1600 A4=INT(A3)
1610 A5=A3-A4
1620 A6=A5*60
1630 A7=INT(A6)
1640 E1=INT(ELE)
```

```
1650 E2=ELE-E1
1660 E3=E2*6Ø
1670 E4=INT(E3)
1680 E5=E3-E4
1690 E6=E5*6Ø
1700 E7=INT(E6)
1710 CALL CLEAR
1720 DISPLAY AT(2,1):" Der S
eitenwinkel(Azimuth)"
1730 DISPLAY AT(3,1):"
                  betr(gt:"
1740 DISPLAY AT(5,6):A1;"$";
A4;"%";A7;"&"
1750 DISPLAY AT(7,1):"    Nac
h Kompa~ einrichten!"
1760 DISPLAY AT(9,1):"@@@@@@
@@@@@@@@@@@@@@@@@@@@@@"
1770 DISPLAY AT(12,1):"Der H
|henwinkel(Elevation)"
1780 DISPLAY AT(13,1):"
                    betr(gt:"
1790 DISPLAY AT(15,7):E1;"$"
;E4;"%";E7;"&"
1800 DISPLAY AT(17,1):"Wasse
rwaage und Winkelmesser"
1810 DISPLAY AT(19,1):"@@@@@
@@@@@@@@@@@@@@@@@@@@@@"
1820 DISPLAY AT(24,1):" Weit
ere Berechnung? <J/N>"
1830 CALL KEY(Ø,W,X)
1840 IF W=74 OR W=1Ø6 THEN 4
7Ø
1850 IF W=78 OR W=11Ø THEN 1
9ØØ
1860 IF X=Ø THEN 183Ø
1870 GOTO 182Ø
1880 CALL CLEAR
1890 GOTO 183Ø
1900 CALL CLEAR
1910 END
```

```
10 ! **********************
11 ! *                    *
12 ! * VERTEILUNGSRECHNUNG *
13 ! *                    *
14 ! *    Copyright by    *
15 ! *                    *
16 ! *    Jochen Zeier    *
17 ! *                    *
18 ! * Benoetigte Geraete *
19 ! *  TI99/4A  Konsole  *
20 ! *     Ext. Basic     *
21 ! *                    *
22 ! *  Speicherbelegung  *
23 ! *     3358 Bytes     *
24 ! *                    *
25 ! **********************
26 !
100 CALL CLEAR :: CALL SCREE
N(5):: ON WARNING NEXT :: FO
R I=Ø TO 14 :: CALL COLOR(I,
16,1):: NEXT I
110 ! DIMENSIONIEREN
120 DIM A(1Ø),L(1Ø),AA(1Ø),G
L(1Ø)
130 ! VORSPANN
140 DISPLAY AT(5,6):"VERTEIL
UNGSRECHNUNG";TAB(6);"======
============="
150 DISPLAY AT(1Ø,13):"VON"
160 DISPLAY AT(14,9):"------
-------";TAB(9);"-HAMSTISOFT-
";TAB(8);"---------------";TA
B(8);"-JOCHEN ZEIER-";TAB(8)
;"---------------"
170 DISPLAY AT(24,1):">IRGEN
DEINE TASTE DRUECKEN!<"
180 CALL KEY(Ø,K,S):: IF S T
HEN 2ØØ ELSE 18Ø
190 ! MENUE
200 CALL CLEAR
210 DISPLAY AT(1,1Ø):"M E N
U E";TAB(1Ø);"========="
220 DISPLAY AT(6,1):"1...NOR
MALE VERTEILUNGS-        REC
HNUNG"
230 DISPLAY AT(9,1):"2...VER
TEILUNGSRECHNUNG BEI     PER
SONENGESELLSCHAFTEN"
240 DISPLAY AT(13,1):"3...EN
DE"
250 DISPLAY AT(23,5):"IHRE W
AHL =>1"
260 ACCEPT AT(23,17)BEEP SIZ
E(-1)VALIDATE("123"):WAHL
270 ON WAHL GOTO 28Ø,58Ø,82Ø
280 !
290 ! NORMALE
    ! VERTEILUNGSRECHNUNG
300 !
310 G,S=Ø :: FOR I=1 TO 1Ø :
: A(I),L(I),AA(I),GL(I)=Ø ::
 NEXT I
320 CALL CLEAR
330 DISPLAY AT(1,1):"NORMALE
 VERTEILUNGSRECHNUNG =======
==================="
340 DISPLAY AT(4,1):"ZU VERT
EILENDER              BETRAG
      : ØØØØØØØØ.ØØ" :: CA
LL E(5,8,S)
350 DISPLAY AT(7,1):"ZAHL D.
BEGUENS-             TIGTEN(
1-1Ø)    :        ØØ" :: ACCEP
T AT(8,24)BEEP SIZE(-2)VALID
ATE(DIGIT):V
360 IF V=Ø OR V>1Ø THEN 35Ø
370 FOR I=1 TO V :: DISPLAY
AT(1Ø+I,1):"ANTEILE VON   ";C
HR$(I+64);" : ØØØØØØØØ.ØØ"
380 CALL E(1Ø+I,8,A(I)):: NE
XT I
```

```
390 DISPLAY AT(23,1):"WURDEN
 VOR- ODER NACHLEIS-  TUNGEN
 GELEISTET          : N"
400 ACCEPT AT(24,28)BEEP SIZ
E(-1)VALIDATE("JN"):WAHL$
410 IF WAHL$="J" THEN 430
420 FOR I=1 TO 10 :: L(I)=0
:: NEXT I :: IF WAHL=2 THEN
650 ELSE 480
430 CALL HCHAR(11,1,32,448)
440 FOR I=1 TO V :: DISPLAY
AT(10+I,1):"LEISTUNG.AN ";CH
R$(I+64);" :V 00000000.00" :
: ACCEPT AT(10+I,16)BEEP SIZ
E(-1)VALIDATE("VN"):L$
450 CALL E(10+I,8,L(I)):: IF
 L$="N" THEN L(I)=-L(I)
460 NEXT I
470 IF WAHL=2 THEN 650
480 ! BERECHNUNG
490 FOR I=1 TO 10 :: S=S+L(I
):: NEXT I
500 FOR I=1 TO 10 :: G=G+A(I
):: NEXT I
510 FOR I=1 TO 10 :: AA(I)=I
NT(S*A(I)/G*100+.5)/100 :: G
L(I)=AA(I)-L(I):: NEXT I
520 ! AUSGABE
530 CALL HCHAR(4,1,32,672)
540 FOR I=1 TO V :: DISPLAY
AT(6+I,1)SIZE(14):"AUSZAHLUN
G  ";CHR$(I+64):: DISPLAY AT
(6+I,16):USING "= ########.#
#":GL(I):: NEXT I
550 DISPLAY AT(24,1):"WEITER
E BERECHNUNGEN(J/N): J"
560 ACCEPT AT(24,28)SIZE(-1)
BEEP VALIDATE("JN"):WAHL$
570 IF WAHL$="J" THEN 200 EL
SE 820
580 !
590 ! VERTEILUNGSRECHNUNG
    ! BEI PERSONENGESELL-
    ! SCHAFTEN
600 !
610 S,ZINS,G=0 :: FOR I=1 TO
 10 :: A(I),L(I),GL(I),AA(I)
=0 :: NEXT I
620 CALL CLEAR
630 DISPLAY AT(1,1):"VERT.BE
I PERSONENGES.-SCHAFT=======
===================="
640 GOTO 340
650 CALL HCHAR(11,1,32,448)
660 DISPLAY AT(12,1):"VERZIN
SUNG DES              STAMMK
APITALS ZU :  000.00 %" :: A
CCEPT AT(13,21)BEEP SIZE(-3)
VALIDATE(DIGIT):GE
670 ACCEPT AT(13,25)SIZE(-2)
VALIDATE(DIGIT):KE :: ZINS=G
E+KE/100
680 ! BERECHNUNG
690 FOR I=1 TO 10 :: S=S+L(I
):: NEXT I
700 FOR I=1 TO 10 :: G=G+A(I
):: NEXT I
710 Q=G*ZINS/100 :: IF Q>=S
THEN 750
720 FOR I=1 TO 10 :: GL(I)=I
NT(A(I)*ZINS+.5)/100-INT(L(I
)*100+.5)/100 :: S=S-INT(A(I
)*ZINS+.5)/100 :: NEXT I
730 FOR I=1 TO 10 :: GL(I)=G
L(I)+INT(S/V*100)/100 :: NEX
T I
740 GOTO 760
750 ZINS=100*S/G :: FOR I=1
TO 10 :: GL(I)=INT(A(I)*ZINS
+.5)/100-INT(L(I)*100+.5)/10
0 :: NEXT I
760 ! AUSGABE
770 CALL HCHAR(4,1,32,672)
780 FOR I=1 TO V :: DISPLAY
AT(6+I,1)SIZE(14):"AUSZAHLUN
G  ";CHR$(I+64):: DISPLAY AT
(6+I,16):USING "= ########.#
#":GL(I):: NEXT I
790 DISPLAY AT(24,1):"WEITER
E BERECHNUNGEN(J/N): J"
800 ACCEPT AT(24,28)SIZE(-1)
BEEP VALIDATE("JN"):WAHL$
810 IF WAHL$="J" THEN 200 EL
SE 820
820 !
830 ! ENDE
840 !
850 CALL CLEAR
860 END
870 !
```



TI REVUE

Das Magazin für TI 99-4A

Nr. 5/87

Einführung in GPL

Der Ton macht die Musik

Bestimmtes Integral

Dialog

Füllstandsmessungen mit dem 99/4A

Turbo und Maschinensprache E/A-Modul

DAS HEFT IM HEFT FÜR TI-USER

EINFACH HERAUSTRENNEN

Grüß Gott -

Wer da glaubt, für den TI gebe es keine Einsatzmöglichkeiten mehr, irrt sich sehr gewaltig. Das beste Beispiel dafür ist unser mehrseitiger Beitrag über den 99er als Kontrollinstrument für den Heizöl-Vorrat. Bevor Sie fragen, was eigentlich Heizöl im Sommer soll?: Gerade im Sommer, denn für dieses Thema sind doch etliche Arbeiten notwendig, die besser vor Beginn der Heizperiode erledigt werden sollten. Denn dann funktioniert's, wenn in der Praxis auf den Knopf gedrückt wird, reibungslos. Ebenso reibungslos ist auch — wie Sie sehen — in diesem Monat wieder ein geschlossenes Heft für alle noch verbliebenen TI-User fertiggestellt worden. Ihnen allen noch einmal zur Erinnerung: Sie brauchen lediglich den Heftmittelteil — wir halten uns da sehr genau daran —, aus der HCA herauszureißen und haben Ihre TI-REVUE. Nicht möglich dagegen ist der Vorschlag eines Lesers, diese bestimmten TI-Seiten sollten etwa fünf Millimeter schmaler und niedriger gestaltet werden. Das geht — zur Zeit — aus technischen Gründen nicht, da wir „am Stück" 48 oder 64 Seiten drukken. Aber wir werden diesen Vorschlag im Auge behalten, diese fünf Millimeter sind sowieso nur Rand. Wie Sie unschwer beim Durchblättern feststelllen können: Wir haben mal wieder Ihre TI-REVUE vollgepackt mit vielem Wissenswerten. Und: Unser R. Ziegler, Ihnen allen bekannt als ausgefuchster Experte, ist wieder da. Nach einer Phase der beruflichen Orientierung — und die geht wirklich vor — kann er sich wieder mit dem heißgeliebten TI beschäftigen. Sie werden von ihm hören. Ist das eine Drohung? Bis zum nächsten Heft Ihre TI-Crew

IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Geschäftsführer: Werner E. Seibt
Redaktion: Alfons Bauer, Thomas Dehn, Jürgen Puschmann, Uwe Schröder
Verantwortlich für den Inhalt:
MSX-Teil: Werner E. Seibt
TI-Teil: Oliver Huber (Technik) Hartmut König (Listings)
Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d
Alle: Postfach 1161
8044 Lohhof.
Anfragen bitte nur schriftlich
Druck: Dierichs, Kassel
Es gilt die Honorarliste des Verlages.
Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.
Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

© by CA Verlag, Heßstraße 90, 8000 München 40 SPS und Autoren

Erstverkaufstag: 24.6.1987

TI99/4A

PREISSENKUNG BEI VIELEN ARTIKELN!!

Peripherie

RS 232 Karte (Atronic)	249,–
32 K-Karte (Atronic)	249,–
Compact Peripherie System CPS 99 mit 1 Diskettenlaufwerk DSDD + Statistikmodul	999,–
CPS 99 mit 2 Diskettenlaufwerken DSDD + Statistikmodul	1299,–
Diskettenlaufwerk DSDD mit Einbausatz für Peripheriebox	329,–
2 Diskettenlaufwerke DSDD mit Einbau- und Kabelsatz f. P.box	679,–
Disccontroller DSDD (Corcomp)	479,–
Externe 32 K-Erweiterung	149,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle	249,–
Externe 32 K-Erweitrung mit Centronicsschnittstelle + Druckerkabel + Epsondrucker LX 800	889,–
dto. + Epsondrucker FX 800	1299,–
dto. + Commodoredr. MPS 1000	969,–
dto. + Stardrucker NL-10	1019,–
80-Zeichen-Karte (Mechatronic)	449,–
GRAM-Karte (Mechatronic)	499,–
Ext. V-24-Schnittstelle (2xRS 232) mit durchgef. Bus + Statistik	149,–
MBX-Sprachsteuereinheit + Sewermania + Baseballmodul anschlußfertig	279,–
WIPO-Mouse 99	199,–
RGB-Modulator mit Scartstecker	179,–
RGB-Modulator + Farbmonitor Commodore AMIGA 1081	969,–
Akkustikkoppler Dataphon S 21 d + externe V-24-Schnittstelle + Verbindungskabel	399,–
Akustikkoppler Dataphon S 21 d + Kabel für RS 232 Karte	279,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter)	115,–
Modulexpander 3-fach	125,–
Fernbedienung (Original TI)	69,–

Commodore/Schneider

Commodore SX-64 (Executive)	1479,–
Plus 4 + Floppy-Disk VC 1551	499,–
Commodore C 128 D	1099,–
AMIGA 500 + Farbmonitor 1081	1969,–
AMIGA 2000 + Farbmonitor 1081	3199,–
Schneider 6128 Grünmonitor	729,–
Schneider 6128 Farbmonitor	1189,–
Schneider PC 1512 mit Schwarzweißmonitor + 1 Laufwerk	1349,–
dto. + 2 Laufwerke	1769,–
PC 1512 mit Farbmon. + 1 Laufwerk	1769,–
dto. + 2 Laufwerke	2199,–
Grafiktablett Supersketch	99,–
Supersketch + Defender + Statistik + Dig Dug + Donkey Kong	169,–

Software

Extended Basic II Plus	229,–
Mini Memory	149,–
Mini Memory + Assemblerbuch dt.	199,–
Editor/Assembler (32 K + Disc)	159,–
Multiplan (32 K + Disc notw.)	199,–
TI-Writer (32 K + Disc notw.)	199,–
TI-Logo II (32 K notwendig)	179,–
Diskfixer (Navarone)	99,–
Terminal Emulator II	79,–
Alpiner, Chisholm Trail, Othello, Car Wars, Tombstone City	je 29,–
Defender, Dig Dug, Invaders, Munchman, Moonmine, Statistik	je 39,–
Fathom, Parsec, Indoor Soccer	je 49,–
Moonsweeper, Microsurgeon, Espial, Burgertime, Donkey Kong, Shamus	je 59,–
Buck Rogers, Video Chess	je 69,–
Chess + Defender + Dig Dug	99,–
Jungle Hunt + Donkey Kong	69,–
Shamus + Donkey Kong + Defender	99,–
Buck Rogers + Moon Patrol + Microsurgeon	129,–
Alpiner + Parsec + Burgertime	89,–
TI-Artist (Diskette)	99,–
Turbo Pascal (Diskette)	149,–
Window-Manager für Turbo Pascal	99,–
C'99-Compiler (Diskette)	99,–

Komplettübersicht über weitere Module + Modulpakete in unserer Preisliste !!!

Bücher

Anwender-Handbuch Turbo Pascal dt.	59,–
Editor/Assembler Handbuch dt.	98,–
TMS 9900 Assemblerhandbuch für das Mini Memory dt.	78,–
TI-99/4A Intern dt.	38,–
Assembler-Kurs (99er-Journal)	59,–
Introduction Assembly Language	38,–

Versandkostenpauschale (Warenwert bis DM 1000,–/darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–).
Lieferung nur gegen NN oder Vorauskasse; Ausland nur Vorauskasse.
Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstrße 5, 7324 Rechberghausen, Tel. (07161) 5 28 89

EINFÜHRUNG IN GPL TEIL 4

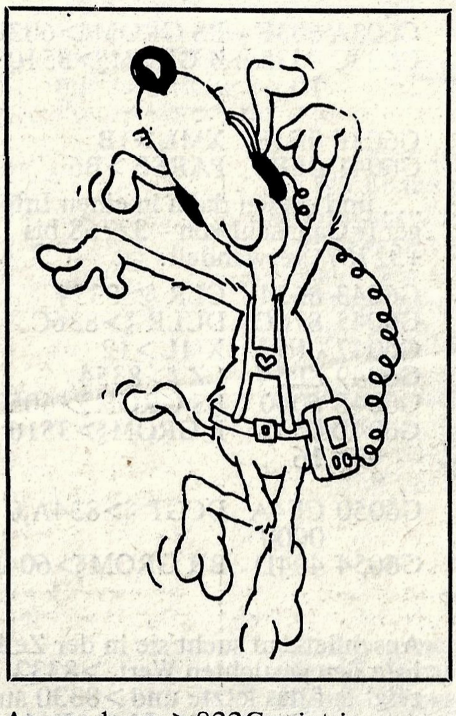
Diesmal gibt es GPL für BASIC-Fans und unser heutiges Demoprogramm ist dementsprechend nicht vom Titelbild her aufrufbar, sondern vom BASIC aus mittels CALL. Es erweitert den BASIC-Interpreter nämlich um zwei Befehle, aber dazu später mehr. Ich muß gleich betonen, daß der BASIC-Interpreter ein sehr komplexes Gebilde ist, das ich im Rahmen einer GPL-Einführung kaum in seiner Funktion darstellen kann, außer eben einige spezielle Eigenschaften.

XML RUFT ROM-ROUTINEN AUF

Im Maschinencode gibt es die Routine XMLLNK. Sie macht genau das, was XML in GPL zu tun hat: Über Tabellen ROM-Routinen aufrufen. Nach XML steht ein Byte, dessen ersten und letzten vier Bits getrennt zu betrachten sind. Mit den ersten vier Bits sucht sich der TI in einer Tabelle, die ab >0CFA im ROM steht, einen von 16 Pointern aus, der auf eine weitere Tabelle zeigt. In dieser wiederum findet der Interpreter mit Hilfe der letzten vier Bits eine von ebenfalls 16 Adressen heraus, die auf die gewünschte Maschinenroutine weist. Sagen wir, der Befehl heißt XML >06. Aus der ersten Tabelle holt er damit den 1. Pointer, der auf eine Auflistung aller gleitkommaverarbeitenden Routinen zeigt, und in dieser zweiten Tabelle schließlich findet er an 7. Stelle die Startadresse der Routine für Gleitkommaaddition, die er daraufhin ausführt. Sie finden alle wichtigen Routinen im Handbuch zum Ed/Ass oder zur Mini-Mem beschrieben, deshalb zeige ich Ihnen hier nur zwei, die wir nachher benötigen und dort nicht stehen:

XML >1B LIEST BASIC-PROGRAMME BYTE FÜR BYTE

Diese für Basic-Anwendungen verwendbare Befehl liest ein Byte aus dem VDP-RAM und legt es in >8342 (Byte) ab. Die Adresse des Bytes muß in >832C (Wort) stehen und wird automatisch erhöht. Ist >8389 (Byte) ungleich Null, so wird das Byte aus dem GROM geholt.



Anwendung: >832C zeigt immer auf das Byte in einem BASIC-Programm, welches als nächstes interpretiert werden soll. Da es auch Module gibt, die BASIC-Programme in ihren GROMs enthalten, dient >8389 als Flag, der in einem solchen Falle ungleich Null wird.

XML >F0 FÜR EIGENE MASCHINENPROGRAMME

In der ersten Tabelle steht an 16. Stelle der Wert >8300. Wenn Sie in >8300 die Startadresse eines vorher geladenen Maschinenprogrammes ablegen und XML >F0 durchführen lassen, so kommen Sie von GPL wieder in die Maschinenebene und Ihr Programm läuft. Der Rücksprung in GPL erfolgt mittels B *R11 (Sie werden z.B. nach einem Execute in EASY BUG die angegebene Adresse immer in >8300 wiederfinden).

PARSE >B6: Kompliziert und äußerst nützlich.

Wer die ROM-Routinen zu PARSE verstehen will, der braucht viel Geduld. Ich selbst habe bisher leider nicht herausfinden können, was der Wert nach PARSE für eine Bedeutung hat. Ich kann Ihnen nur sagen, was bei PARSE >B6 geschieht: >832C muß auf eine Angabe eines numerischen Wertes in einem BASIC-Programm zeigen. Diese Angabe kann eine Gleitkommazahl, eine Variable oder ein ganzer Rechenausdruck sein. Z.B.: 26+A*1.07

Die Befehlsfolge XML >1B, PARSE >B6 bewirkt dann, daß dieser Ausdruck gelesen und sein Wert im sog. RADIX 100-Format in den 8 Bytes >834A bis >8351 abgelegt wird. Außerdem erfolgt eine automatische Korrektur von >832C auf das diesem Ausdruck folgende Byte, das zudem noch in >8342 kommt.

Bisher noch unbekannt: EXEC, CONT und RTNB

Diese drei speziellen BASIC-Anweisungen konnte ich trotz bestem Bemühen bisher nicht entschlüsseln. Deshalb hier die Aufforderung an andere GPL-Programmierer, an dieser Stelle einmal darüber zu schreiben. Für mich sind diese Anweisungen jedenfalls zu hoch . . .

ADRESSOPERANDEN: SONDERFORMATE

Wenn Sie diese Folge lesen, kennen Sie bereits alle Operandenformen des GPL. Mit zwei Ausnahmen: *>837C und Zugriffe auf >837D. Es sind Sonderformate, die speziell abgefragt werden und die ich der Einfachheit halber bisher unterschlagen habe.

XIII. PUSH UND *>837C

Mit dem Befehl PUSH können Sie Werte auf den DATA-Stack legen und z.B. mit ST ...,*>837C wieder lesen. Dieses Verfahren können Sie in der kommenden Folge detailliert nachlesen.

XIV. ZUGRIFFE AUF >837D

Hier besteht die Besonderheit darin, daß mit dem Aufruf erst ein Wert überhaupt in >837D geschrieben wird. Es ist ein Byte vom Bildschirm, dessen Adressierung über >837E (YPT) und >837F (XPT) geschieht. Auch dieses lernen Sie erst in der

kommenden Folge richtig kennen, da ich dann erst auch Verwendung für diese Formate habe.

DAS PROGRAMM: CALL GOTOX UND CALL GOSUBX

Sie wissen wohl, daß Sie im BASIC nur zu fest angegebenen Zeilennummern springen können. Selbst ON X GOTO schafft da nur bedingt Abhilfe. Einziger Vorteil: Sie können mit RES die Zeilennummern beliebig umsortieren, ohne daß Veränderungen am Programm selbst nötig sind. Hier bekommen Sie nun zwei Routinen, die eigentlich nichts weiter als eine andere Schreibweise der Befehle GOTO und GOSUB sind, nur daß jede beliebige Form der Zeilenangabe möglich ist.
Was halten Sie denn davon:

```
100 J=-1
110 CALL GOSUBX(210+J)
120 J=-J
130 Q=Q+1
140 CALL GOTOX(INT(Q/12)
    *40+110)
150 END

200 PRINT "HALLO";
210 RETURN
220 PRINT "DU DA!"
230 RETURN
```

Damit so etwas möglich ist, müssen die Routinen GOTOX und GOSUBX aber erst definiert sein.
Bisher hatten wir im Header den Pointer immer an die 4. Stelle gesetzt. Diesmal muß er an die 6. Stelle (siehe Beschreibung in der 1. Folge):

```
G6000 AA00  0000 0000 0000Header
G6008 0000  6010 0000 0000

G6010 601A  6032 0547 4F54
G6018 4F58             "GOTOX"

G601A 0000  6025 0647 4F53
G6022 5542  58         "GOSUBX"
```

Bei G6025 startet die Routine GOSUBX. Sie unterscheidet sich von GOTOX nur dadurch, daß vor dem Sprung die aktuelle Zeilennummer auf den sog. Basic-Value-Stack kommt. Er liegt im VDP-RAM und >836E zeigt auf den letzten Eintrag. Jeder dieser Einträge ist 8 Bytes lang. Der Wert >66 als 3. Byte kennzeichnet den Eintrag als RETURN-Adresse, die in den ersten beiden Bytes liegt. Genauer gesagt ist es nicht die Adresse selbst, sondern der Pointer auf die aktuelle Stelle in der Zeilenliste, der dort liegt.
Value-Stack-Pointer hochzählen

```
G6025 A36E  DADD §>836E,8          Value-Stack-Pointer
      0008                         hochzählen
G6029 BDB0  DST VDP*>836E,§>832E   Zeilenpointer
      6E2E                         auf Stack
G602D BEE0  ST VDP§>0002(§>836E),>66  Identifier
      026E                         Return Adress auf
      66                           Stack
```

Bei G6032 startet die Routine GOTOX, die sich zunächst einmal die im BASIC-Programm angegebene Sprungadresse als Gleitkommazahl im RADIX-100-Format (es ist im Handbuch zum Ed/Ass erklärt!) holt . . .

```
G6032 BD2C  DST §>832C,§>8356      Pointer auf aktuelles
      56                           Byte im BASIC-Prg. holen
G6035 0F1B  XML>1B                 Byte holen
G6037 D642  CEQ §>8342,>B7
      B7
G6035 0F1B  XML

G603A 603F  BS GROM§>603F          Ja: Weiter
G603C 0535  B GROM§>3510           Nein: Error "Incorrect
      10                           Statement" ausgegeben,
                                   da kein Wert vorhanden
G603F 0F1B  XML >1B
G6041 0EB6  PARSE >B6              Wert holen
```

. . . und diesen dann in einen Integer (=Ganzzahl von −32768 bis +32767) verwandelt:

```
G6043 8654  CLR §>8354             Errorcode löschen
G6045 876C  DCLR §>836C
G6047 0F12  XML >12                Routine zum Umwandeln
G6049 8E54  CZ §>8354              Error aufgetreten?
G604B 6040  BS GROM§>6050          Nein: Weiter
G604D 0535  B GROM§>3516           Ja: Error "Bad Value"
      16                           ausgeben, da Zahl zu
                                   groß oder zu klein
G6050 CF4A  DCGT §>834A,0          Wert negativ oder
      0000                         gleich Null?
G6054 404D  BR GROM§>604D          Ja: Error "Bad Value"
                                   ausgeben
```

Anschließend sucht sie in der Zeilenliste den gesuchten Wert. >8332 zeigt auf das letzte und >8330 auf das erste Byte in der Liste, die immer abwechselnd eine Zeilennummer (Wort) und die Startadresse der zugehörigen Zeile im VDP-RAM enthält (auch Wort).

```
G6056 BD10  DST §>8310,§>8332      Pointer kopieren...
      32
G6059 9110  DINC §>8310            ...und auf erstes Byte
                                   nach der Liste ändern
G605B A710  DSUB §>8310,4          Hier beginnt die Such-
      0004                         schleife
G605F D5B0  DCEQ VDP*>8310,§>834A  Wert in Liste
      104A                         gefunden
G6063 6085  BS GROM§>6085          Ja: Schleife beenden
G6065 D530  DCEQ §>8330,§>8310     Liste vollständig
      10                           durchsucht?
G6068 405B  BR GROM§>605B          Nein: Weiter
G606A 0628  CALL GROM§>284E        Ja: Error ausgeben
      4E
G606D 6072  DATA >6072             Textpointer
```

```
G606F 0520   B GROM§>2012          Return to Basic
      12
```

Wir wollen uns an dieser Stelle eine eigene Errormeldung schaffen. In G606A steht der Aufruf der Errorroutine. Sie benötigt einen Pointer auf den auszugebenden Text, der hier auf G6072 steht. Dort findet sie unseren Text, der mit Screen-Offset und Längenpointer versehen so aussieht:

```
G6072 12BA  A5A9  ACA5  80B5  AEA1  "ZEILE
G607C B5A6  A6A9  AEA4  A2A1  B2    UNAUFFINDBAR"
```

Damit haben wir eine deutsche Fehlermeldung. Auch so etwas ist in BASIC möglich!

```
G6085 D642   CEQ §>8342,>B6        Letztes von PARSE ge-
      B6                           lesenes Byte ")"?
G6088 403C   BR GROM§>603C         Nein: Error "Incorrect
                                   Statement", da Klammer
                                   nicht geschlossen
G608A 0F1B   XML >1B               nur zur internen
                                   Pointerkorrektur
```

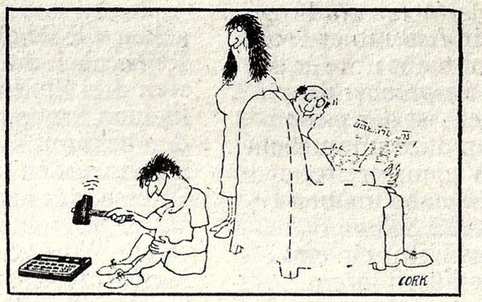
Letztentlich müssen wir den BASIC-Pointer >832E (siehe Beschreibung zur GOSUBX-Routine) für die gewünschte neue Zeile setzen. Er zeigt auf die Stelle in der Zeilenliste, die die Startadresse einer Zeile angibt (nicht auf deren Nummer!). Sie dürfen ihn jedoch nicht auf die gewünschte Zeile setzen, sondern auf die nächst niedrigere, da der Interpreter nach dem Rücksprung ja davon ausgeht, daß dort noch der Pointer auf die aktuelle Zeile liegt und ihn daraufhin selber auf die folgende Zeile hochzählt. Da die Zeilenliste überdies noch rückwärts im VDP-RAM liegt, müssen wir den Pointer aber nicht herunter-, sondern heraufzählen; um vier Bytes, damit er auf die nächstniedrigere Zeile zeigt und um weitere zwei Bytes, damit er nicht auf die Zeilennummer, sondern auf die Adresse weist:

```
G608C A310   DADD §>8310,6         vorhergehende Zeile
      0006
G6090 BD2E   DST §>832E,§>8310     Pointer setzen
      10
G6093 06000  CALL GROM§>0012       GSR-Return
      12
```

Und nun viel Spaß und Programmiererfolg mit den neuen Routinen! Der Vollständigkeit halber muß aber noch gesagt werden, daß GOSUBX nicht ganz „wasserdicht" ist. Es legt nämlich die RETURN-Adresse auf den Value-Stack, ohne zu prüfen, ob dort überhaupt noch Platz ist. Bei langen Programmen kann es so mit der Zeit zu Überschneidungen mit der Variablenliste kommen, was ein ziemliches Chaos zur Folge hätte. Für eine wirklich ernsthafte Anwendung dieser Routine müssen Sie dann mit XML >17 arbeiten, das Werte auf den Stack legt und dabei eine vollständige Errorabfrage durchführt. Der Übersicht wegen habe ich in diesem Programm darauf verzichtet. *Sven Dyroff*

TI REVUE

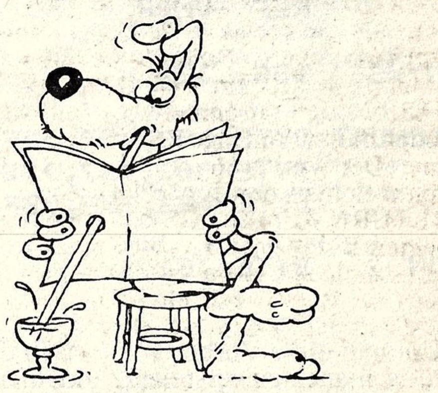
DAS SUPER-SONDER ANGEBOT



Unser Kassetten-Service räumt sein Lager! Wir bieten an: Bunte Programm-Mischung bestehend aus drei oder fünf Kassetten unterschiedlichen Inhalts mit Spielen, Utilities, Anwender-Programmen. Nur solange Vorrat reicht. Zu Super-Preisen: Das Dreier-Paket für sage und schreibe DM 30, das Fünferpaket für ganze 40 DM! Keine Rücknahme, keine Wunschzusammenstellung! Benutzen Sie unseren Bestell-Coupon im Innern des Heftes, einfach ankreuzen, Geld, Scheck beifügen oder Bankabbuchung. Ist die Ware vergriffen, wird der Scheck nicht eingelöst, beiligendes Bargeld zurückerstattet.

TI-BESTELL-COUPON

AUF SEITE 49



DER TON MACHT DIE MUSIK

Zwar nicht gerade zu Beethoven, aber doch zu einem recht passablen Philharmoniker kann man seinen 99er erziehen. Gewußt wie! TI REVUE sagt es Ihnen hier

Um ein Assembler-Programm nicht nur für das Auge ansprechend zu machen, ist der passende Ton dazu unumgänglich. Vielen bereitet es aber beinahe unüberwindliche Schwierigkeiten, auch nur ein einziges Geräusch ihrem Computer zu entlocken. All denen soll dieser Artikel dienen. Man kann danach zwar noch keine Sinfonien den Computer komponieren lassen, aber er kann sie jedoch schon spielen.
Um auf den Soundgenerator zugreifen zu können, sind, wie immer, einige Vorbereitungen notwendig.

```
* **********************************
* ***********************************
* *                                 *
* * Sounderzeugung mit dem TI-99/4A *
* *                                 *
* *   Beispielprogramm by O.Huber   *
* *                                 *
* ***********************************
* **********************************

       DEF   TON           STARTNAME 'TON'

       REF   VMBW

MYWS   BSS   >20           EIGENER WS

TONDAT BYTE  >03,>89,>3F,>9E,>78  DATEN FIR DEN SOUND
       BYTE  >03,>A7,>32,>93,>B3,>78
       BYTE  >02,>9F,>BF,>00

ON     BYTE  >01           'TESTBYTE'
```

Um nun diese Vorbereitungen auch zu verstehen, gehen wir sie einfach der Reihe nach durch. Die ersten drei Zeilen dieses kleinen Programmes werden bekannt sein. Mit ihnen werden die Startadresse, das benötigte Unterprogramm und der eigene Workspace definiert. Interessant dagegen werden die angegebenen DATAs. In ihnen sind alle Angaben enthalten, die der Computer benötigt, um dem Zweiklang A-C# zu spielen. Die parallelen Basic-Daten hierzu wären CALL SOUND (120,110,6)::CALL SOUND(120,139,6).
Der Unterschied besteht aber momentan noch darin, daß der Computer nach dem Ausführen des kleinen Assemblerprogrammes noch keinen Ton von sich gibt, doch immer der Reihe nach.
Gehen wir zunächst auf die angegebenen Daten ein:
Byte 1: Es gibt an, wieviele Bytes der Computer für die Erzeugung des Tones verwenden soll.
In unserem Fall sind es immer drei Bytes, da nur ein Ton gleichzeitig angesteuert werden soll.
Bytes 2 und 3: Sie geben die Frequenz des Tones an und den Tongenerator, auf dem er gespielt werden soll. Eine Liste der Frequenzen befindet sich im E/A-Handbuch auf Seite 318ff. Die höchstwertigen vier Bit

FREQUENZ UND DÄMPFUNG

geben dabei den Soundgenerator an. '8'= Soundgenerator #1, 'A' = Soundgenerator #2 und 'C' = Generator #3. Die letzten Bit bestimmen die Frequenz.
Byte 4: Mit diesem Byte wird indirekt die Lautstärke des Tones angegeben. Es wird dem Computer mitgeteilt, um wieviel Dezibel der Ton gedämpft werden soll.
Es steht hierbei '>9x' für Generator #1, '>Bx' für #2, '>Dx' für #3 und '>Fx' für den Rauschgenerator.
Das 'x' steht für den Wert der Dämpfung, also mit wieviel Dezibel der Ton gedämpft werden soll. Dieser berechnet sich folgendermaßen. Im Byte handelt es sich um die Bits 5–8. Es wird aber bei der Berechnung im Computer ein Bit mehr verwendet. Das heißt, es wird einfach eine Null hinten rangehängt. Hat man also die Dämpfung b1101 eingegeben, so wertet sie der Computer als b11010 aus. Es wird also der doppelte Wert verwendet. Will man nun eine Dämpfung von 2 Dezibel erreichen, so muß man b0001 angeben, das dann vom Computer zu b00010 umgewandelt wird, und den Wert 2 hat.
Byte 5: Dieses Byte gibt die Länge des Tones an. Man gibt ihn in 1/60 sec an. In unserem Beispiel sind es also >78/60=120/60= 2 sec.
Die Bytes, die die Frequenz und die Dämpfung angeben, sind immer gleich aufgebaut. Bei der Frequenz handelt es sich um ein 16Bit-Wort, wobei das erste Byte angibt, daß eine Frequenz

SOUNDLISTE AUFBAUEN

folgt. Dieses angebende Byte ist sowohl bei der Frequenz als auch bei der Dämpfung folgendermaßen aufgebaut:
Bit 1: Immer '1'
Bit 2–4: Anweisung
Be Frequenz: Bit 5–8: Nybble des Frequenzwertes.
Bei Rauschen: Bit 5: Weißes oder periodisches Rauschen
Bit 6–8: Art des Rauschens.
Bei Dämpfung: Bit 5–8: Wert der Dämpfung.
Die Anweisungen, die in den Bits 2–4 möglich

sind, heißen folgendermaßen:

0 0 0 Generator #1, Frequenz
0 0 1 Generator #1, Dämpfung
0 1 0 Generator #2, Frequenz
0 1 1 Generator #2, Dämpfung
1 0 0 Generator #3, Frequenz
1 0 1 Generator #3, Dämpfung
1 1 0 Rauschgenerator, Art des Rauschens
1 1 1 Rauschgenerator, Dämpfung.

Die letzte Zeile der DATAs bewirkt die Abschaltung der Tongeneratoren. Es wird hierbei nur die maximale Dämpfung der einzelnen Generatoren hintereinander angegeben. '9F' für Generator #1, 'BF' für Generator #2, 'DF' für Generator #3 und 'FF' für den Rauschgenerator. Als Länge wählt man dann >00.

Die Bedeutung des Bytes unter dem Label 'ON' soll uns an dieser Stelle noch nicht interessieren. Um die Vorbereitungen abzuschließen, muß noch im VDP eine Soundliste aufgebaut werden. Dies geschieht dadurch, daß man die unter 'TONDAT' angegebenen Daten an eine beliebige Stelle ins VDP kopiert.

Sind diese Vorbereitungen korrekt abgeschlossen, so kommt erst die eigentliche Soundverarbeitung. Diese will ich an der Fortführung meines kleinen Programmes erläutern.

INTERRUPT UNTERBINDEN

Um zunächst die Soundabarbeitung zuzulassen, muß der Interrupt unterbunden werden. Dies geschieht mittels dem Befehls 'LIMI 0'. Ist dies vollendet, so setzt man das Wort auf Adresse >83CC auf die Adresse, auf der die eigene Soundliste im VDP liegt, also auf deren Beginn im VDP. Im Anschluß daran setzt man das sogenannte VDP-RAM-Flag. Dies geschieht dadurch, daß man auf Adresse >83FD das höchstwertige Bit in den Zustand '1' setzt. Hier zeigt sich schon die Verwendung des Bytes unter dem Label 'ON'. Dieses Byte dient auch dazu, die Soundbearbeitung zu starten. Man muß dazu das höherwertige Byte der Adresse >83CE auf den Wert >01 setzen. Anschließend wird der Interrupt mittels 'LIMI2' wieder zugelassen und der Rechner beginnt zu tönen. Woher weiß man aber nun, wann die gesamte Soundliste abgearbeitet ist? Nun, das ist ganz einfach. Ist die Soundliste abgearbeitet, wird das höherwertige Byte von Adresse >83CE auf >00 zurückgesetzt. Es genügt also ein einfacher Vergleich gegen Null, um das Ende zu „erwischen". Der letzte Teil des Beispielprogrammes stellt einen einfachen Return dar, der, in diesem Fall, ins E/A-Modul zurückführt. Erweitert man nun seine Sound-Liste, so darf man auf keinen Fall vergessen, auch entsprechend mehr Bytes ins VDP-RAM zu übertragen. Es kann sich sonst ein übles Erwachen aus den schönen, harmonischen Dreiklängen ergeben.

Bei den weiteren Durchführungen will ich nur noch von den DATAs reden, da das Grundprogramm im Grunde das gleiche bleibt.

Das vorgestellte Programm spielt nun 2 sec den Kammerton A, dann zusätzlich den Ton C# und kehrt nach weiteren 2 sec ins Basic zurück.

Will man nun aber die beiden Töne hintereinander erklingen lassen, so hat man zwei Möglichkeiten. Entweder verwendet man nur einen Tongenerator. Dieser stellt dann um auf die neue Frequenz und es ist nur ein Ton zu hören. Die andere Möglichkeit ist, den einen Generator

DATAS VERTAUSCHEN

einfach vorzeitig abzustellen. Hierzu muß man nur die DATAs untereinander vertauschen.

Sie sehen dann folgendermaßen aus:

TONDAT BYTE >03, >8E, >0F, >93, >78
BYTE >04, >AA, >0C, >B3, >9F, >78
BYTE >01, >BF, >00

Jetzt wird der vorhergehende Tongenerator gleichzeitig abgeschaltet, wenn der neue Ton erzeugt wird. Damit sind wir schon im neuen Thema, dem Ansteuern aller Tongeneratoren zur selben Zeit.

In Basic existiert hier ja die einfache Aneinanderreihung der Werte im CALL SOUND. Ähnlich funktioniert dasselbe auch in Assembler. Betrachten wir einfach die Basic-Zeile CALL SOUND (120,440,0,554,2,659,4). Die parallelen DATA-Zeilen in der Soundliste sehen folgendermaßen aus:

TONDAT BYTE >09, >8E, >0F, >AA, >0C, >CA, >0A, >90, >B1, >D2, >78
BYTE >3, >9F, >BF, >DF, >00

Byte 1: Anzahl der zu

```
       EVEN

TON    LWPI MYWS            EIGENER WS LADEN
       LI   Ø,>1ØØØ         VDP ADRESSE
       LI   1,TONDAT        SOUND-DATEN
       LI   2,>ØØØF
       BLWP @VMBW           INS VDP

LOOP   LIMI Ø               INTERRUPT VERBIETEN
       LI   1Ø,>1ØØØ        VDP-ADRESSE IN R1Ø
       MOV  1Ø,@>83CC       ZEIGER ZUR TAFEL SETZEN
       SOCB @ON,@>83FD      VDP RAM FLAG SETZEN
       MOVB @ON,@>83CE      SOUNDBEARBEITUNG STARTEN
       LIMI 2               INTERRUPT ZULASSEN

LOOP1  MOVB @>83CE,@>83CE     SOUND ABGEARBEITET?
       JNE  LOOP1             NEIN

EXIT   CLR  1               BASIC-RETURN
       MOVB 1,@>837C
       LWPI >83EØ
       B    @>ØØ7Ø

       END
```

übertragenden Bytes.
Bytes 2–3: Frequenz des Tones A, Generator #1.
Bytes 4–5: Frequenz des Tones C#, Generator #2.
Bytes 6–7: Frequenz des Tones E, Generator #3.
Byte 8: Dämpung von #1 um 0 dB
Byte 9: Dämpfung von #2 um 2 dB.
Byte 10: Dämpfung von #3 um 4 dB.
Byte 11: Dauer >78/ 60 sec= 2 sec.
Die zweite DATA-Zeile bewirkt allein das Abschalten der Generatoren. Dies geschieht durch die Angabe der Dämpfung von 30 dB für jeden Generator und die Dauer von 0 sec. Der richtige Sound ist jetzt also nur noch dem eigenen Gehör zu verdanken.
Zur Dämpfung der Generatoren ist noch zu sagen, daß man sie einfach „weiterlaufen" lassen kann. Es genügt dann, beim nächsten Ton eines anderen Generators die neue Dämpfung anzugeben. Die Frequenz behält er dann, nur mit neuer Lautstärke, bei.
Im Beispiel sieht das so aus:

```
TONDAT BYTE >03,
>89,>3F,>9E,>78
       BYTE >04,
>AA,>0C, >95,>B5,
>78
       BYTE >02,
>9F,>BF,>00
```

Man kann sich dadurch sparen, Frequenzen nochmals anzugeben, wenn sie beibehalten werden sollen.
Nach dieser Einführung sollte es nun dem Einzelnen möglich sein, ein musikalisches Werk den Rechner spielen zu lassen.

DAS RAUSCHEN

Was bisher ganz vernachlässigt wurde, ist der Rauschgenerator. Dieser ist aber bei einem Spiel von unbedingter Notwendigkeit.
Gehen wir wieder von einer Basic-Zeile aus.
CALL SOUND (120, –3,0) erzeugt in Basic ein ca. 2 sec langes periodisches Rauschen vom Typ 3. In Assembler sieht die dazu parallele DATA-Liste so aus:

```
TONDAT BYTE >2,
>E9,>F0,>78
       BYTE >01,
>FF,>00
```

Die einzelnen Bytes bewirken:
Byte 1: Anzahl der zu übergebenden Bytes.
Byte 2: Bestimmung des Rauschgenerators und des Rauschtyps. Dabei sind die einzelnen Bits wie folgt belegt:
Bit 1: Ist immer '1'.
Bit 2–4: '1 1 0', damit wird dem Computer mitgeteilt, daß es sich um den Rauschgenerator handelt. Speziell für die Erzeugung eines Rauschens.
Bit 5: '1' = periodisches Rauschen
'0' = weißes Rauschen.
Bit 6–8: Art des Rauschens. Anhand folgender Tabelle kann man ersehen, welche Werte welches Rauschen erzeugen.

Bits 6–8	Basic-Rauschen
0 0 0	–1
0 0 1	–2
0 1 0	–3
0 1 1	–4
1 0 0	–5
1 0 1	–6
1 1 0	–7
1 1 1	–8

Byte 3: Es legt die Lautstärke fest, mit der das Rauschen ertönen soll. Die Bits sind dann folgendermaßen belegt:
Bit 1: Ist immer '1'
Bits 2–4: ' 1 1 1 ' dieser Wert steht für die Dämpfung der Rauschgenerators.
Bit 5: Ist immer '0'
Bits 6–8: Größe der Dämpfung. Sie berechnet sich wie die Dämpfung eines normalen Tones.
Nun kann einer wirklichen Sinfonie mit Schlagzeug nichts mehr im Wege stehen. Was bei dieser Soundliste jedoch immer noch bleibt, ist die mühsame Eingabe der Frequenzen. Ich kann dabei nur viel Spaß wünschen.

FÜR SPEZIALISTEN

Es besteht ja, wie hoffentlich bekannt, beim TI die Möglichkeit, in Assembler den Interrupt zu unterbinden. Es kann folglicherweise auch passieren, daß man trotz vollkommen unterbundenem Interrupt Sound erzeugen will.
Bisher mußte man doch immer irgendwann durch 'LIMI 2', den Interrupt wieder zulassen. Es geht aber auch ohne!
Zunächst muß dem betreffenden Assembler-Programm die Direktive 'REF SOUND' hinzugefügt werden. Danach ist eine kleine Änderung an der Soundliste notwendig. Bei dieser Art von Sounderzeugung gibt es keine Längenangabe des Tones mehr. Dieses Byte muß also entfernt werden. Wie lange ein Ton dauert, bestimmt man selbst. Die Generatoren „laufen" so lange, bis man sie durch die Dämpfung '>xF' abschaltet. Hat man nun seine Soundliste im Reinformat, so muß man sie nicht mehr im VDP ablegen. Es kann ja auch möglich sein, daß eventuell das VDP belegt ist, z.B. durch den Bit-Map-Mode etc. Es genügt nun vollauf, die Soundliste auf Adresse >8400 im CPU-RAM, zu legen. Dies muß natürlich Ton für Ton erfolgen, da er sie nicht mehr der Reihe nach abarbeitet, sondern den Ton anspielt, sobald seine Daten komplett sind.
Spreche ich nun von einem Ton, so meine ich dabei die Daten-Anzahl der zu übergebenden Bytes, Frequenz(en) und der(en) Dämpfung(en).
Wie genau das Ablegen der Töne auf Adresse >8400 erfolgt, zeigt folgendes Beispiel:

```
        REF  SOUND
        DEF  TON

STATUS  EQU   >837C
GPLWS   EQU   >83E0
TONADR  EQU   >8400
NEXT    EQU   >0070
TONDAT  BYTE  >03,>8C,>1F,>91
TONEND  BYTE  >01,>9F
MYWS    BSS   >20

TON     LWPI  MYWS
        LI    R1,TONDAT
        LI    R2,TONADR
        MOVB  *1+,*2
        MOVB  *1+,*2
        MOVB  *1+,*2
        MOVB  *1,*2
        LI    R3,>FFFF

LOOP    DEC   R3
        JNE   LOOP

        LI    R1,TONEND
        MOVB  *1+,*2
        MOVB  *1,*2

EXIT    CLR   R1
        MOVB  *1, STATUS
        LWPI  >83E0
        B      NEXT

        END
```

Anhand der MOVB-Befehle wird die Soundliste auf Adresse >8400 geladen. Wie zu sehen ist, beginnt sie nicht ab Adresse >8400, sondern wird Byte für Byte auf diese Adresse geladen. Die Warteschleife dient nur dazu, daß der Ton eine Weile zu hören ist.
Dies war alles, was zur Einführung in die Welt der Töne zu sagen ist.
Ich hoffe, vielen damit geholfen zu haben und wünsche einen guten Ton mit dem TI.

Oliver Huber

FARBMONITOR AN DEM TI

Bitte beantworten Sie mir folgende Frage zum TI. Ist es möglich und sinnvoll, einen Farbmonitor an den TI anzuschließen? Wie kann man dieses „vollbringen"?
Hans-Jürgen Klöhn, Lampertheim

Ein Kabel, das einen Farbmonitor mit Composite-Video-Eingang an den TI anschließt, wurde in TI-REVUE 7/85 beschrieben. Um einen Farbmonitor mit RGB-Eingang anzuschließen, ist ein RGB-Modulator notwendig.

BESTIMMTES INTEGRAL

Als erstes fragt der Computer die Funktion f(x) ab. Diese wird dann mittels „Zeilenpoke“, welches APARAK5 aus TI-REVUE 9/85 entnommen ist, ins Programm geschrieben. Von dieser nun neu entstandenen DEF-Zeile geht das Programm aus und nimmt sie als grundliegende Funktion.
Die Lösungsvariable heißt 'X'. Es ist nicht möglich, weitere Parameter innerhalb der Funktion zu verwenden. Nach der Angabe der Funktion gibt man das Intervall an, innerhalb welchem integriert werden soll. Der linke Rand ist hier die „untere Grenze“, der rechte Rand die „obere Grenze“.
Der Rechner unterteilt nun dieses Intervall in 1000 Stellen, die die gleiche Entfernung voneinander haben. Die Funktionswerte dieser Stellen werden nun in der Variablen W() zwischengespeichert.
Die Unterteilung des Intervalls und der an der jeweiligen Stelle dazugehörige Funktionswert bilden ein annähernd genaues Trapez. Die Summe aller dieser Trapeze strebt gegen das erste bestimmte Integral.
Nach der Berechnung der Summe erfolgt die Berechnung des arithmetischen Mittels.
Will man dann noch das quadratische Mittel berechnet haben, so muß man diesen Punkt einfach anwählen. Es ist hierbei jedoch zu beachten, daß ein neues Integral errechnet werden muß. Dies hat einen Zeitverlust zur Folge.
Das Programm selbst integriert natürlich nicht jede beliebige Funktion. Sie muß natürlich auch den Gesetzen der Integralrechnung unterliegen. In diesem ist die Funktion f(x) innerhalb des abgeschlossenen Intervalls (x1,x2) integrierbar, wenn sie
- in (x1,x2) beschränkt mit nur endlich vielen Unstetigkeitstellen,
- in (x1,x2) stetig, und monoton ist.

```
80 REM *******************
81 REM *                 *
82 REM *   BESTIMMTES    *
83 REM *    INTEGRAL     *
84 REM *                 *
85 REM *      VON        *
86 REM *                 *
87 REM *  THORSTEN REHM  *
88 REM *                 *
89 REM *******************
90 REM SPEICHERBEREICH VON
     9460-9677
91 REM STARTADRESSE 9530
100 OPTION BASE 1 :: DIM W(1
000):: CALL INIT :: CALL SCR
EEN(2):: CALL CLEAR :: GOTO
120
110 DEF Y(X)=X^3 :: RETURN :
: !2 :: RETURN :: !!AAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAA
120 CALL CHAR(36,"0008141010
10101010101010101010101010
10105020000000000000000000")
130 FOR I=0 TO 8 :: CALL COL
OR(I,16,2):: NEXT I
140 CALL HCHAR(2,3,45,28)::
CALL HCHAR(22,3,45,28):: DIS
PLAY AT(1,1):"BESTIMMTES INT
EGRAL"
150 FOR I=9460 TO 9677 :: RE
AD A :: CALL LOAD(I,A):: NEX
T I
160 CALL LOAD(8196,63,248)::
 FOR I=1 TO 8 :: READ A$ ::
CALL LOAD(16375+I,ASC(A$))::
 NEXT I
170 DATA 0,0,0,1,0,0,0,0,0,0
180 DATA 0,1,203,20,203,53,2
03,78,203,231
190 DATA 204,71,37,184,204,2
28,205,29,205,75
200 DATA 205,96,2,62,89,47,
47,81,78,18
210 DATA 0,0,0,0,0,0,0,0,204
,112
220 DATA 204,112,204,112,204
,112,4,91,4,192
230 DATA 204,64,204,64,204,6
4,204,64,4,91
240 DATA 2,224,36,244,2,1,37
,20,6,160
250 DATA 37,46,4,192,2,1,0,1
,4,32
260 DATA 32,12,2,0,131,74,2,
1,37,20
270 DATA 6,160,37,36,2,1,37,
28,6,160
280 DATA 37,46,2,5,3,233,6,5
,2,1
290 DATA 0,2,4,192,192,5,4,3
2,32,12
300 DATA 2,0,37,20,2,1,131,9
2,6,160
310 DATA 37,36,4,32,32,24,14
,136,2,0
320 DATA 37,28,2,1,131,92,6,
160,37,36
330 DATA 4,32,32,24,13,128,2
,0,131,74
340 DATA 2,1,37,28,6,160,37,
36,2,133
350 DATA 0,1,22,222,2,0,37,2
8,2,1
360 DATA 131,74,6,160,37,36,
4,192,2,1
370 DATA 0,1,4,32,32,8,4,224
,131,124
380 DATA 2,224,131,224,4,96,
0,112
390 DATA I,N,T,E,G,R,%,:
400 !
410 ! EINGABE F(X)
420 !
430 DISPLAY AT(4,1):"F(X)="
:: ACCEPT AT(4,6):FX$ :: FQ$
="("&FX$&")^2" :: FP$=FX$&"
DX" :: CALL ZP(FX$,110):: GO
SUB 110
440 !
450 ! EINGABE UG-OG
460 !
470 DISPLAY AT(6,1):"UNTERE
GRENZE:" :: ACCEPT AT(6,15):
UG :: UG$=STR$(UG):: IF LEN(
UG$)=1 THEN UG$=UG$&".0"
480 DISPLAY AT(8,1):"OBERE G
RENZE:" :: ACCEPT AT(8,14):O
G :: OG$=STR$(OG):: IF LEN(O
G$)=1 THEN OG$=OG$&".0"
490 IF OG<=UG THEN 470 ELSE
FK=(OG-UG)/999 :: Z=0
500 !
510 ! ABTASTUNG&RECHNUNG
520 !
530 FOR X=UG TO OG STEP FK :
: Z=Z+1 :: W(Z)=Y(X):: NEXT
X :: CALL HCHAR(23,3,32,28):
: CALL LINK("INTEGR",FK,W())
540 !
550 ! DISPLAY
560 !
570 CALL HCHAR(3,1,32,608)::
 CALL HCHAR(5,4,36):: CALL H
CHAR(6,4,37):: CALL HCHAR(7,
4,38):: DISPLAY AT(4,1):UG$
:: DISPLAY AT(8,1):OG$
580 DISPLAY AT(6,3):FP$ :: D
```

```
ISPLAY AT(1Ø,3):"="|FK :: AM
=(1/(OG-UG))*FK :: DISPLAY A
T(12,1):"ARITHM.MITTEL="|AM
59Ø DISPLAY AT(14,1):"QUADRA
TISCHES MITTEL ? (J/N)"
6ØØ CALL KEY(Ø,K,S):: IF S=Ø
 THEN 6ØØ ELSE IF K=74 THEN
61Ø ELSE 63Ø
61Ø CALL ZP(FQ$,11Ø):: GOSUB
 11Ø :: Z=Ø :: FK=(OG-UG)/99
9 :: FOR X=UG TO OG STEP FK
:: Z=Z+1 :: W(Z)=Y(X):: NEXT
 X :: CALL LINK("INTEGR",FK,
W())
62Ø QM=SQR((1/(OG-UG)*FK)::
 DISPLAY AT(14,1):"QUAD.MITT
EL="|QM
63Ø DISPLAY AT(16,1):"NEUE F
KT.? (J/N)"
64Ø CALL KEY(Ø,K,S):: IF S=Ø
 THEN 64Ø ELSE IF K=74 THEN
CALL HCHAR(3,1,32,6Ø8):: GOT
O 43Ø ELSE STOP
65Ø SUB ZP(FX$,ZN):: CALL PE
EK(-3195Ø,A,B):: EE=A*256+B-
65537
66Ø CALL PEEK(-31952,A,B)::
SS=A*256+B-65536
67Ø CALL PEEK(EE-2,A,B):: IF
 A*256+B<>ZN THEN EE=EE-4 EL
SE 69Ø
68Ø IF EE>SS THEN 67Ø ELSE P
RINT "ZEILE NICHT DA" :: STO
P
69Ø CALL PEEK(EE,A,B):: CC=A
*256+B-65536
7ØØ CALL PEEK(CC,R):: IF R<>
131 AND R<>154 AND R<>137 TH
EN PRINT "KEIN REM" :: STOP
71Ø CALL LOAD(CC=137,89,183,
88,182,19Ø):: CC=CC+6
72Ø DATA ABS,ATN,COS,EXP,INT
,LOG,SGN,SIN,SQR,TAN
73Ø PP=1
74Ø FOR AA=1 TO LEN(FX$)
75Ø B=ASC(SEG$(FX$,AA,1))
76Ø IF B=43 THEN DD=193 ELSE
 IF B=45 THEN DD=194 ELSE IF
 B=42 THEN DD=195 ELSE IF B=
47 THEN DD=196
77Ø IF B=4Ø THEN DD=183 ELSE
 IF B=41 THEN DD=182
78Ø IF B=94 THEN DD=197
79Ø IF "PI"=SEG$(FX$,AA,2)TH
EN DD=221 :: AA=AA+1 :: GOTO
 91Ø
8ØØ RESTORE 72Ø
81Ø FOR M=1 TO 1Ø
82Ø READ B$
83Ø IF B$=SEG$(FX$,AA,3)THEN
 CALL LOAD(CC+PP-1,M+2Ø2)::
AA=AA+2 :: GOTO 93Ø
84Ø NEXT M :: Q=1
85Ø IF B<>46 AND B<48 OR B>5
7 THEN 9ØØ
86Ø IF (B<>43 OR B<>45)AND A
A>2-Q THEN IF SEG$(FX$,AA+Q-
2,1)="E" THEN 88Ø
87Ø IF B<>46 AND B<48 OR B>5
7 AND B<>69 THEN 88Ø ELSE Q=
Q+1 :: IF LEN(FX$)<Q+AA-1 TH
EN 88Ø ELSE B=ASC(SEG$(FX$,A
A+Q-1,1)):: GOTO 86Ø
88Ø Q=Q-1 :: E$=SEG$(FX$,AA,
Q):: CALL LOAD(CC+PP-1,2ØØ,Q
):: FOR AAA=1 TO Q :: CALL L
OAD(CC+PP+AAA,ASC(SEG$(E$,AA
A,1))):: NEXT AAA
89Ø CC=CC+AAA :: AA=AA+Q-1 :
: GOTO 93Ø
9ØØ IF B>64 AND B<91 THEN DD
=B
91Ø IF DD=Ø THEN PRINT "EING
ABEFEHLER" :: STOP
92Ø CALL LOAD(CC+PP-1,DD)
93Ø PP=PP+1 :: NEXT AA
94Ø CALL LOAD(CC+PP-1,13Ø,13
6,13Ø,131):: SUBEND
```

```
* ASSEMBLERPROGRAMM ZU BESTIMMTES INTEGRAL
* BY THORSTEN REHM

* AUFRUF IBER: CALL LINK("INTEGR",FK,W())

* BEI START: FK =SCHRITTWEITE
*            W()=FUNKTIONSWERTE

* BEI ENDE:  FK =FLICHENINHALT

        DEF   INTEGR

NUMREF  EQU   >2ØØC
NUMASG  EQU   >2ØØ8
XMLLNK  EQU   >2Ø18
FMUL    EQU   >ØE88
FADD    EQU   >ØD8Ø
NEXT    EQU   >ØØ7Ø
STATUS  EQU   >837C
GPLWS   EQU   >83EØ

FAKTOR  BSS   >Ø8
SUMME   BSS   >Ø8
MYREGI  BSS   >2Ø

        EVEN

SCHIEB  MOV   *Ø+,*1+
        MOV   *Ø+,*1+
        MOV   *Ø+,*1+
        MOV   *Ø+,*1+
        RT
```

```
CLEAR  CLR  0
       MOV  0,*1+
       MOV  0,*1+
       MOV  0,*1+
       MOV  0,*1+
       RT

INTEGR LWPI MYREGI           EIGENER WS
       LI   1,FAKTOR
       BL   @CLEAR           FAKTOR=0
       LI   1,>0001
       BLWP @NUMREF          FK HOLEN
       LI   0,>834A          FAC
       LI   1,FAKTOR
       BL   @SCHIEB          FAKTOR IN FAC
       LI   1,SUMME
       BL   @CLEAR           SUMME=0
       LI   5,>03E9          R5=1001

LOOP   DEC  5
       LI   1,>0002          2te VARIABLE
       MOV  5,0              INDEX->R0
       BLWP @NUMREF          HOLEN
       LI   0,FACTOR
       LI   1,>835C          ARG
       BL   @SCHIEB          FAKTOR->ARG
       BLWP @XMLLNK          W()*FAKTOR
       DATA FMUL             MULTIPLIZIEREN
       LI   0,SUMME
       LI   1,>835C          ARG
       BL   @SCHIEB          SUMME->ARG
       BLWP @XMLLNK
       DATA FADD             ARG+FAC
       LI   0,>834A          FAC
       LI   1,SUMME
       BL   @SCHIEB          (ARG+FAC)->FAC
       CI   5,1              W(1)?
       JNE  LOOP             NEIN, NOCHMAL

       LI   0,SUMME
       LI   1,>834A          FAC
       BL   @SCHIEB          SUMME->FAC
       CLR  0                KEIN ARRAY
       LI   1,>0001          1te VARIABLE
       BLWP @NUMASG          FK=SUMME
       MOVB 0,@STATUS        BASIC-RETOUR
       LWPI GPLWS
       B    @NEXT

       END
```

PASCAL AM TI

Ich hätte da folgende Frage zum TI. Ich habe mir neulich die P-Code-Karte für meinen Rechner angeschafft. Dazu besitze ich noch folgendes: Compiler-Disk, E/A-Disk, Assembler/Linker-Disk, E/A-Modul und Line-by-Line-Assembler-Kassette. Nun zu meiner Frage: Was benötige ich noch, um UCSD-Pascal programmieren zu können?

Marcel Estermann, CH-Oberengstringen

Um in UCSD-Pascal programmieren zu können, benötigen Sie noch zwei weitere Programmpakete. Das eine ist der Pascal-Editor, das andere das Pascal Utility-Paket. Ohne diese ist es nicht möglich, Programme in Pascal zu erstellen.

FEHLER IN DER DEBUGGER-HARDCOPYROUTINE?

Bei der Assemblierung der in TI-REVUE 12/86 abgedruckten Hardcopy für den Debugger traten bei mir mehrere Fehler während der Assemblierung auf. Ist es möglich, daß die Hardcopy-Routine fehlerhaft im Heft abgedruckt wurde?

Hermann Homann, Minden

Das Programm wurde, bevor es abgedruckt wurde, getestet. Bei diesen Tests traten keinerlei Fehler auf. Es ist also nur möglich, daß Ihr File einen Fehler hat.

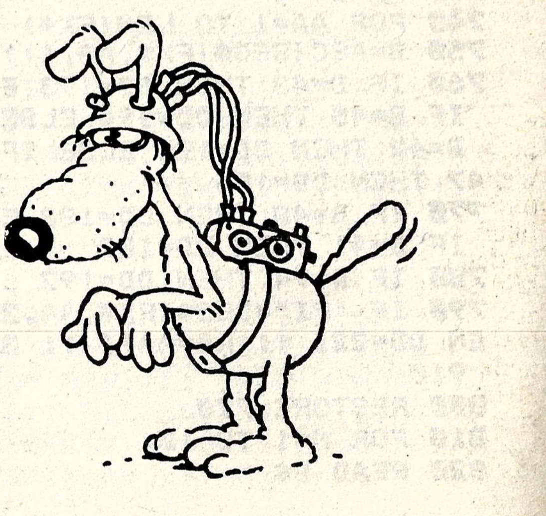
16-BIT SPEICHERERWEITERUNG

In letzter Zeit werden immer mehr selbstgestrickte 32k Speichererweiterungen mit 16 Bit Datenbus für den TI-99 angeboten, die 2 – 3mal schneller als solche mit 8 Bit breiten Datenbus sein sollen.

Als ich meine Unterlagen durchsah, fand ich heraus, daß der TMS9900 zwar 16 Datenleitungen besitzt, jedoch nur 8 auf dem I/O-Port herausgeführt worden sind. Da außerdem die Speicheradressierbarkeit der TMS9900 mit 32k Worten, also mit 64 kBytes, angegeben ist, würde ich jetzt gerne wissen, ob und wie ein Umbau einer Speichererweiterung möglich ist, und ob dadurch Probleme mit z.B. dem Diskettenhandling auftreten können. Ebenfalls würde es mich interessieren, ob sich durch eine solche Speichererweiterung der Speicherplatz verdoppeln ließe.

Georg Ch. Jordan, Brunsdorf

Wie Sie schon erwähnten, sind nur 8 Datenleitungen aus dem Rechner herausgeführt. Daher sind 16-Bit-Erweiterungen nur intern, also im Rechnergehäuse, möglich. Der Speicherplatz kann durch eine solche Erweiterung nicht verdoppelt werden, da der TI, trotz seiner 8 Datenleitungen, 16-Bit verarbeitet. Es bleibt also bei 32x2k-Bytes. Je nach Anschluß läßt sich ein Zeitgewinn von 0–3fach erzielen. Letzterer ist jedoch nur in Assembler möglich. Probleme mit Peripheriegeräten sind nicht bekannt, jedoch laufen einige Module mit einer solchen Erweiterung nicht. Der Grund hierfür ist bisher noch nicht bekannt.



FÜLLSTANDMESSUNGEN MIT DEM TI99/4A

Ohne Frage ist der Computer aus den Büros heute schon nicht mehr wegzudenken. Doch im privaten Bereich gibt es recht wenige sinnvolle Anwendungen. Eine, die nicht gerade naheliegt, kann für Bewohner von Eigenheimen mit Ölheizung „problemlösend" sein, nämlich die hier beschriebene Füllstands-Meßeinrichtung für den Heilzöltank. Computeranwendung wird eigentlich erst interessant, wenn aus einer „Datenerfassung" – und das ist die Feststellung des derzeitigen Ölstandes im Tank – automatisch Schlußfolgerungen gezogen werden sollen. Im Hinblick auf Heizöl ergeben sich immerhin drei Fragen, die Antwort erheischen

Computeranwendungen wird eigentlich erst interessant, wenn aus einer „Datenerfassung" – und das ist die Feststellung des derzeitigen Ölstandes im Tank – automatisch Schlußfolgerungen gezogen werden sollen. Im Hinblick auf Heizöl ergeben sich immerhin drei Fragen, die Antwort erheischen:

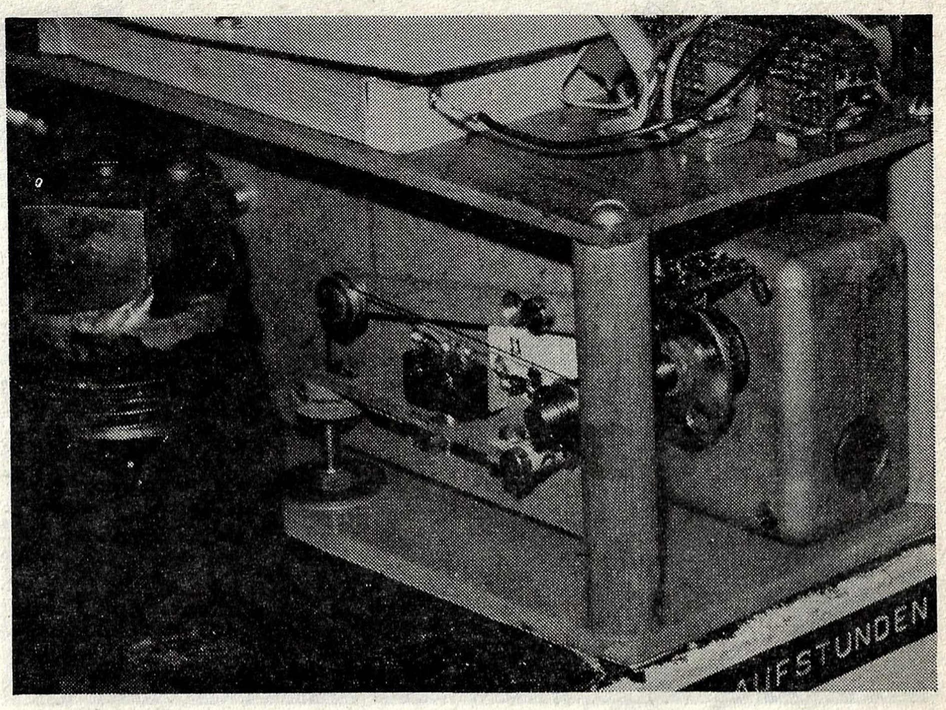
1. Wie groß ist der derzeitige Vorrat?
2. Wie lange reicht er voraussichtlich?
3. Wann ist der richtige Zeitpunkt für die Bestellung?

Klar, wessen Öltank größer ist als der größte Jahresbedarf, braucht darüber nicht nachzudenken. Doch werden oft Tanks mit nur 3000 Liter eingebaut, weil bei denen die Sicherheitsbedingungen billiger zu erfüllen sind. Da muß der Winter schon beinahe ein Frühling sein, wenn das Öl ohne nachzutanken reichen soll.

GELD SPAREN PER COMPUTER

Die Fragen 2 und 3 haben sich im Frühjahr 1986 und 1987 als sehr wichtig erwiesen, denn der Ölpreis sank zu einer Zeit, als er sonst stieg. Mit Computerhilfe ließ sich da durchaus Geld sparen. Denn aus den Füllstandsdaten der laufenden Überwachung ließ sich der tägliche Ölverbrauch und die Verschiebung des jeweils gültigen Bestellzeitpunkts verfolgen. Das milde Wetter schob ihn auf diese Weise immer weiter voraus.

Wenn man solche Überwachung und Berechnung von Hand ausführt, braucht man, mit dem Gang in den Keller, jedesmal 10 Minuten. Mit dem Computer dauert es nur eine Minute und es geht von der warmen Stube aus, so mal eben nebenbei, ohne Füße und Kopf anzustrengen. Das voraussichtlich richtige Bestelldatum war einfach vom Bildschirm abzulesen. – Wenn das keine sinnvolle Anwendung für einen Heimcomputer ist, dann gibt es überhaupt keine! Damit wäre das, was man heutzutage „Motivierung" nennt, gefunden. Jetzt geht es um die technische Durchführung.



1.
Der erste selbstgebaute Füllstandmesser des Autors am „Einsatzort", gleich neben dem Stutzen für den Peilstab. Durch diesen Stutzen mußte die Steckbuchse praktiziert, der Faden geholt und der Schwimmer eingesetzt werden.

Die Technik bietet viele Wege, den Vorrat in einem Behälter festzustellen, doch nur sehr wenige sind für das Selberbauen gangbar. Der einfachste wäre, die Zahl der Betriebsstunden des Brenners oder, mit einem „Wasserzähler", die Menge des verbrauchten Heizöls festzustellen, dafür gibt es nämlich elektrisch „auslesbare" Zähler.

Zweckmäßiger ist es jedoch, eine Meßmethode anzuwenden, die eine Aussage über die wirklich noch im Behälter befindliche Menge macht. Es muß also festgestellt werden, wie hoch das Heizöl noch im Behälter steht, demnach ist eine „Füllstandmessung" nötig.

Hierbei ist es nun unwichtig, ob der Füllstand vom Flüssigkeitsspiegel nach unten zum Boden oder zu einem festen Punkt oberhalb des Spiegels gemessen wird. Die vorhandene Menge kann dann anhand der Form und Abmessungen des Behälters berechnet werden.

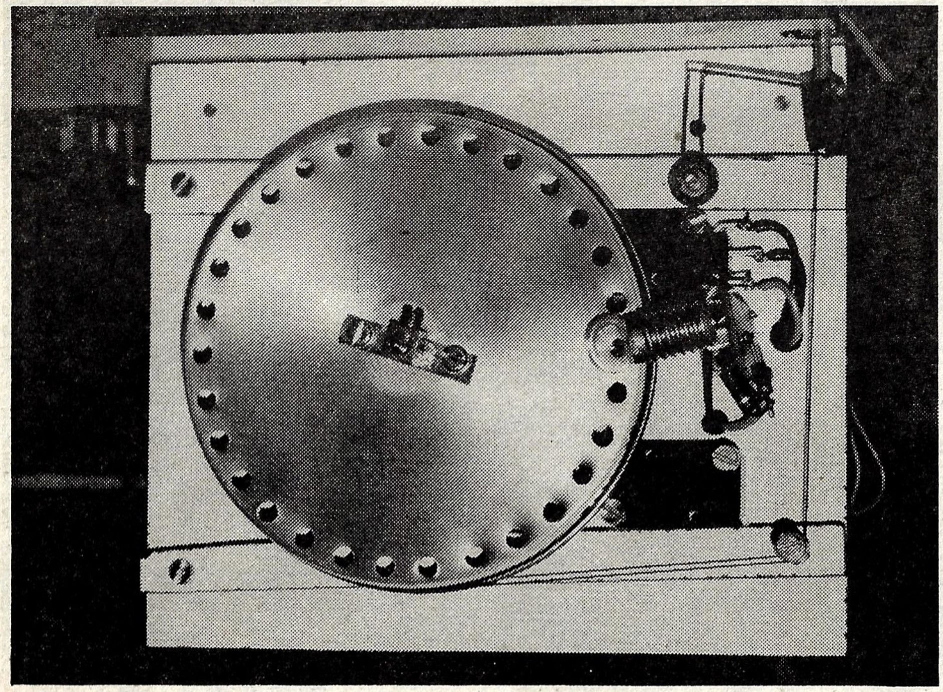
Daraus folgt, das, was gemessen werden soll, ist eine „Länge" und die ist in Meter bzw. in Teilen davon zu messen.

Bekannt ist ja die Füllstandsmessung mit einem „Peilstab", sie ist sehr einfach und wird deshalb sehr viel verwendet. Doch es geht auch umgekehrt, das Maß muß nicht in die Flüssigkeit getaucht werden. Ebensogut läßt sich mit einem Bandmaß von einem festen Punkt oberhalb des Flüssigkeitsspiegels messen. Von dort wird es abgerollt. Sobald der Flüssigkeitsspiegel erreicht ist, braucht das abgelesene Maß nur noch von dem Maß zwischen dem festen Punkt und dem Behälterboden abgezogen zu werden und es kommt das gleiche Maß heraus, wie es der eintauchende Peilstab ergibt. Für die praktische Ausführung braucht es kein Bandmaß, ein Faden genügt. Der Faden bekommt einen Schwimmer, der feststellt, wann der Flüssigkeitsspiegel erreicht ist.

Dank Archimedes wissen wir, daß ein Körper, in Flüssigkeit getaucht, um soviel „leichter" wird wie das Gewicht der Flüssigkeit, die er ver-

drängt. Schwimmt der Körper, so hat er „kein Gewicht". Nur das Gewicht des Fadens ist übrig.
Indem der Faden an einem Waagebalken hängt, der bei Entlastung hoch geht, ist dies ein Signal, daß der Schwimmer auf der Flüssigkeit angekommen ist. Das ist zugleich auch ein „digitales" Signal, denn der Waagebalken hat nur zwei Stellungen:

unten = Schwimmer hängt noch in der Luft
oben = Schwimmer schwimmt.



2.
Der verbesserte und nachbaubare, hier beschriebene Füllstandmesser. Die Seiltrommel mit den Löchern für die Lichtschranke und deren Lichtquelle sind nicht zu übersehen. Ebenso deutlich ist die Anordnung der beiden Mikroschalter, deren Betätigungshebel und der Verlauf des Fadens zu sehen.

DAS SELBSTBAU-FÜLLSTANDSMESSGERÄT

Dazu erst mal eine Vorbemerkung des Autors:
„Hier folgt keine bis in die letzte Einzelheit ausgearbeitete Bauanleitung. Das hat nichts mit Faulheit zu tun, sondern mit meiner Einstellung zum Basteln. Ich halte nichts davon, sich als Bastler oder „Heimwerker" wie ein Fließbandarbeiter zu betätigen, nur, daß der Arbeitsplatz am Küchentisch und nicht die Fabrik ist. Wer daheim und zum eigenen Spaß bastelt, der soll auch ‚kreativ' sein dürfen. Er soll also nicht einer Schablone folgen, sondern eigene Ideen verwirklichen. Eine Bastelanleitung dient zuerst dazu, Anregungen dafür zu vermitteln, was ‚bastelbar' ist. Ein ausgeführtes Muster soll zeigen, daß das Ziel erreichbar und was bei der Verwirklichung zu bedenken ist. Wer eine andere Lösung der einen oder anderen Einzelheit für besser hält, möge sie ausprobieren. Nicht zuletzt darin liegt der Spaß!
Komme mir keiner mit dem Spruch: ‚Nun habe ich es ganz genau nachgebaut – aber es funktioniert nicht!'

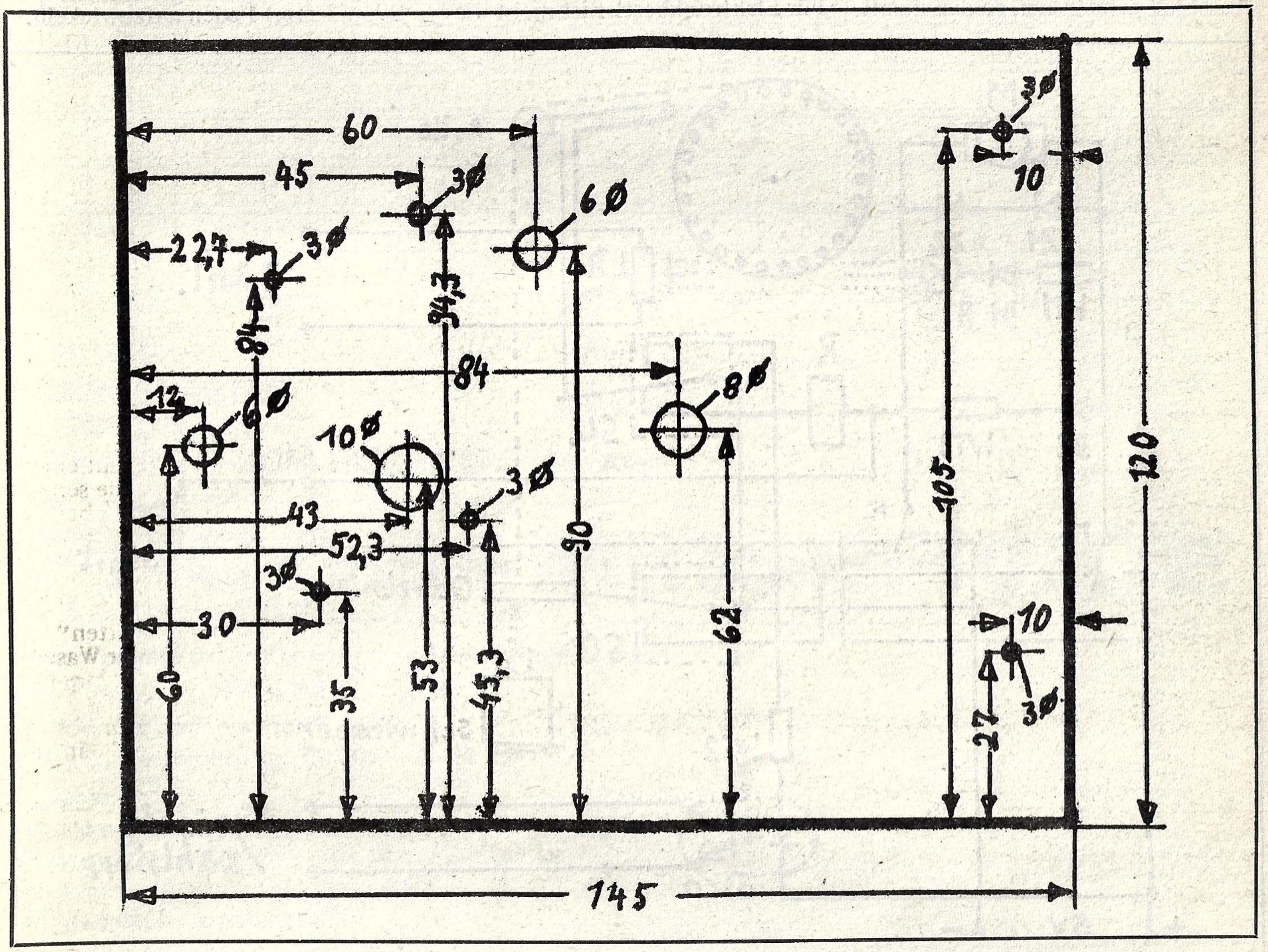
Viel lieber lese ich in Leserbriefen: ‚Ich habe das so und so gemacht, das müßte doch auch gehen – warum geht es nicht?' "
Das beschriebene Arbeitsprinzip kann auf verschiedene Weise technisch verwirklicht werden. Hier sollen aber all die denkbaren Ausführungsmöglichkeiten nicht weiter diskutiert werden.
Das erste Füllstandsmeßgerät des Autors (Bild 1) entstand „aus der Bastelkiste", also aus zufällig vorhandenen Teilen. Das hier beschriebene Gerät (Bild 2) hingegen ist „reproduzierbar", also aus käuflichen Teilen gebaut, so daß der Nachbau an der Teilebeschaffung nicht scheitern kann.
Das Gerät enthält einen Elektromotor, der in beiden Drehrichtungen laufen kann, ein Getriebe, das ins Langsame (etwa 10 Umdrehungen je Minute) untersetzt und eine „Seiltrommel" antreibt, die einen (Polyester-)Zwirnsfaden ab- und aufwickelt. Die Länge des ablaufenden Fadens wird aus den Umdrehungen der Seiltrommel abgeleitet und sinnvoller Weise mit einer Lichtschranke gezählt. Der Waagebalken ist hier ein einarmiger Hebel, der auf einen Mikroschalter aufliegt. Am losen Ende des Hebels ist eine Rolle, über die der Faden läuft. Hängt der Schwimmer am Faden daran, dann ist der Mikroschalter betätigt, das Eigengewicht des Hebels und des Fadens allein reicht dazu nicht aus.
Der Faden läuft aber auch noch durch einen Ring oder eine Gabel an einem zweiten Hebel. Dieser Hebel wird vom Schwimmer angehoben, wenn der Faden fast ganz auf die Trommel aufgewickelt ist und betätigt dann einen zweiten Mikroschalter.
Diese Apparatur liefert also Signale über folgende Zustände:
1. Ruhezustand, Faden aufgetrommelt, Schwimmer oben
2. Faden läuft ab, Schwimmer belastet den Faden
3. Schwimmer unten, schwimmt, Faden schlaff.

Alle Bestandteile werden auf eine Platte (Bild 3) montiert, die senkrecht auf einer Grundplatte befestigt ist. Die Platten können aus beliebigem Material sein. Für das Mustergerät wurde doppelseitig beschichtete Hartfaserplatte verwendet, wie sie als „Kachelplatten" zur Wandbekleidung hinter Waschbecken benutzt wird. Es genügen zwei Stücke von Kachelgröße.
Die wenigen elektronischen Bauelemente finden auf einer kleinen Streifen-Leiterplatte Platz.

SCHALTUNGSBESCHREIBUNG (BILD 4)

Der hier benutzte Elektromotor (M) arbeitet mit Gleichstrom, die Drehrichtung hängt von der Polung seiner Anschlüsse ab. Sie wird von zwei Umschaltekontakten (r) eines (neutralen) Relais (R) bestimmt.
In der Ruhelage, also bei stromloser Relais-Spule, ist der Motor auf „Aufwickeln" geschaltet. Ist der Faden aufgewickelt, also der Schwimmer oben, so wird der Mikroschalter (SO) vom Gabelhebel betätigt und der Motor gestoppt (gezeichnete Schalterstellung von SO und SU).
Damit ist der Ruhezustand des Gerätes automatisch hergestellt, ohne daß der Computer daran beteiligt war.
Um den Meßvorgang auszulösen, muß das Relais (R) Strom bekom-

men. Dafür muß ein (Start-)Taster kurz gedrückt werden.
Hat das Relais (R) Strom, so schalten seine beiden Kontakte (r) den Motor ein. Daraufhin beginnt der Motor die Trommel in Ablaufrichtung zu drehen, der Faden abzulaufen und der Schwimmer sich zu senken. Der abwärts gehende Schwimmer gibt den Gabelhebel frei und der Mikroschalter (SO) die Federkraft des Mikroschalters (SU) den Rollenhebel an und der Kontakt schließt den Stromkreis der Leitung „Freigabe/Zählstopp". Daraufhin bricht das Programm die Zählung ab. Der Stromkreis des Relais (R) wird unterbrochen, seine Kontakte (r) kehren in die Ruhelage zurück und schalten die Drehrichtung des Motors um, so daß der Faden wieder aufgewickelt wird. Infolge der Umpolung der Spannung am Motor erlischt auch die Beleuchtung der Lichtschranke.



3.
Bohrplan für die Haupt-Montageplatte des Füllstandmessers (nur als Beispiel)

öffnet den Stromkreis der Leitung „Freigabe/Zählstopp". Dies muß vom Programm ausgewertet und die Zählung freigegeben werden. Die Beleuchtung (BL) der Lichtschranke wurde mit dem Motor zusammen eingeschaltet. Während der Faden abgewickelt wird, sendet die Lichtschranke (RL) über die „Zähl"-Leitung abwechselnd Hi- und Lo-Signale zum Computer.
Sobald der Schwimmer die Öloberfläche erreicht und schwimmt, wird der Faden schlaff. Infolgedessen hebt

Sobald der Schwimmer am Faden hängt, wirkt sein Gewicht wieder auf den Rollenhebel und dessen Mikroschalter (SO) öffnet einerseits den Stromkreis der Leitung „Freigabe/Zählstopp", dies wird jedoch vom Programm nicht ausgewertet. Andererseits wird der Stromkreis für das Relais (R) wieder vorbereitet. Damit ist der Zustand erreicht, der im ersten Absatz dieses Abschnittes beschrieben ist.
Wenn der Motor wieder stillsteht, fließt ein schwacher Strom durch die Relaisspule (R), der jedoch nicht ausreicht, die Kontakte (r) zu betätigen. (Dieser stört jedoch nur, wenn die Meßeinrichtung aus einer Batterie betrieben wird. Abhilfe wä-

STROMVERSORGUNG

re beispielsweise durch einen zweiten Mikroschalter am Gabelhebel möglich, mit dem der Haltestromkreis des Relais (D2,Wi3) abgeschaltet wird. Diese Möglichkeit ist hier jedoch nicht dargestellt.)
Es wäre natürlich am bequemsten, den Strom für die Meßeinrichtung (d.h. für Relais und Motor) dem Computer zu entnehmen. Jedoch bricht die Spannung durch den Motor-Anlaufstrom zusammen, so daß das Programm ausfällt. Also muß eine eigene Stromversorgung vorgesehen werden.

VERBINDUNG MIT DEM COMPUTER

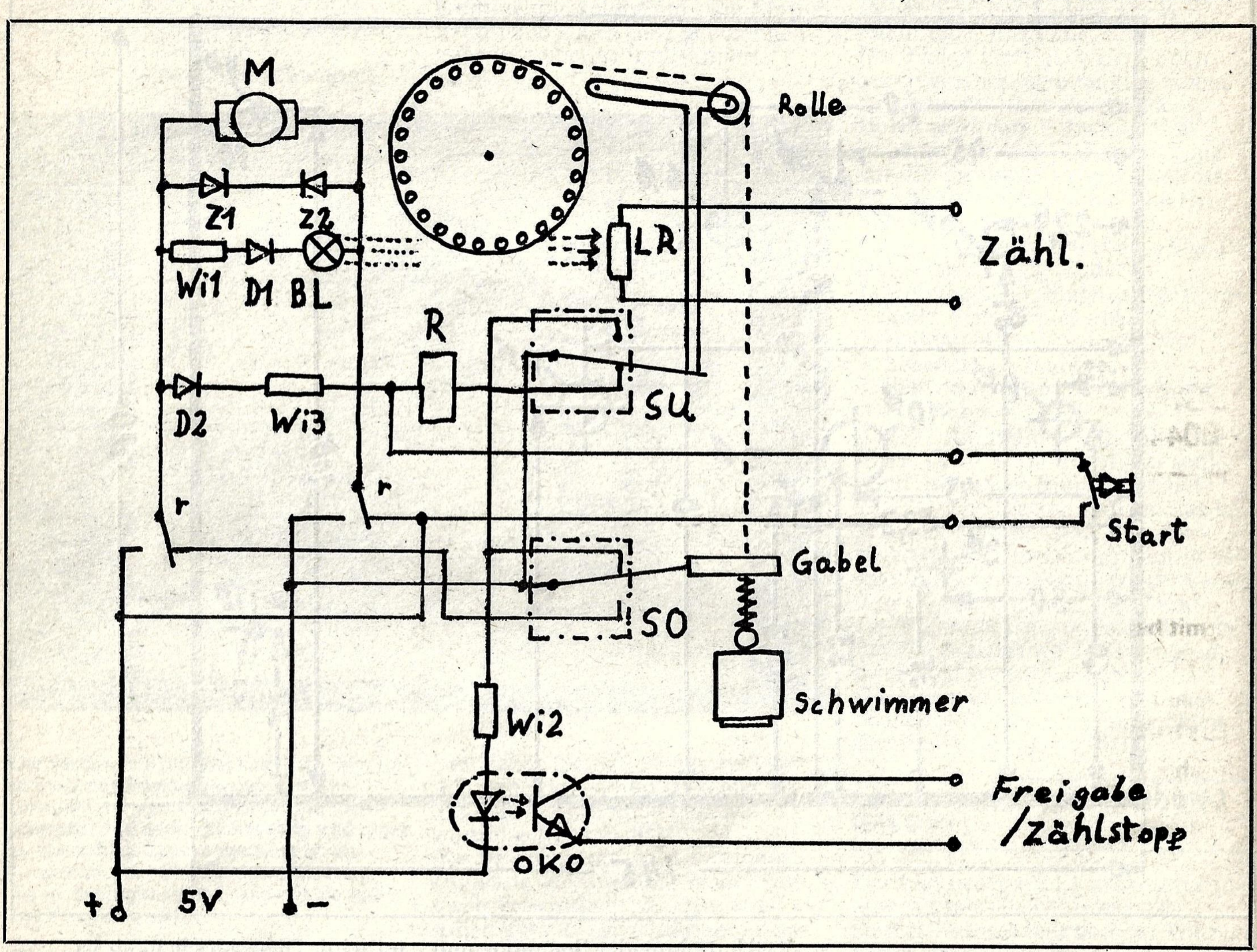
Die „Joystick"-Anschlüsse genügen, um die Signale des Füllstandsmeß-

gerätes dem Computer zur Auswertung zu übergeben.
Joysticks enthalten vier Schalter für die vier Hauptbewegungsrichtungen des Steuerknüppels. (Die schrägen Bewegungen werden aus den Signalen von zwei benachbarten Schaltern zusammengesetzt.) Ein fünfter Schalter sitzt oben im Knüppel als Auslöser oder „Trigger".
Alle diese Kontakte haben einen Anschluß gemeinsam, so daß damit sechs von den neun Anschlüssen des Joystick-Steckers belegt sind. Für TI 99/4A werden hier die Anschlüsse angegeben. Die Füllstand-Meßeinrichtung wird also mit einem Joystick-Stecker an den Computer angeschlossen.

HINWEISE FÜR DEN SELBSTBAU

Für Elektronkbastler ist meist die Mechanik das größte Problem.
Doch die handwerklichen Anforderungen für den Bau des Füllstandmessers sind wohl von jedem Heimwerker zu erfüllen, der eine Bohrmaschine und eine Säge besitzt.
Der Elektromotor mit Getriebe ist als Bausatz erhältlich und läßt sich in 5 Minuten ohne Werkzeug zusammensetzen. Nur die Abtriebswelle muß passend gekürzt werden.
Zum Betrieb ist das Getriebe zu ölen; da das Öl dann beim Lauf herumspritzt, empfiehlt es sich, einen Blechstreifen (Alublech von einer Cremedose) nach Augenmaß zuzuschneiden und herumzulegen.

Für die „Seiltrommel" fand sich kein geeignetes Bauteil in den Bastlerkatalogen, daher ist dafür Selbstbau nicht zu umgehen. Wer eine Drehbank besitzt, wird kaum Schwierigkeiten haben, aber es geht auch ohne: Ein Deckel von der sehr verbreiteten 250g-Hautcremedose hat genau den richtigen Durchmesser, um bei einer Umdrehung 30 cm (genau genommen 295,63 mm) Faden aufzuwickeln.
Da kaum ein Heizöltank höher als zwei Meter ist, sind nur 6 Windungen nötig. Um eine zufriedenstellende Meßgenauigkeit zu erreichen, darf nämlich der Faden nicht mehrlagig aufgewickelt werden, denn jede Fadenlage macht den Wickeldurchmesser und damit die Windungslänge größer.
Um den Füllstand zentimetergenau messen zu können, müssen also je Umdrehung der Seiltrommel 30 Impulse zum Computer gegeben werden. Demnach müssen 30 Löcher für die Lichtschranke gebohrt werden. Das wichtigste Werkzeug hierzu, ebenso zur Bestimmung



4.
Der Schaltplan des Füllstandmessers
Teileliste:

Wi 1	Kohleschichtwiderstand	10 Ohm 0,5 W
Wi 2	dgl.	150 Ohm 0,1 W
Wi 3	dgl.	22 Ohm 0,5 W
D1, D2	Siliziumdiode 1 N 4001	
Z1, Z2	Zehnerdioden 7,5 V, Typ beliebig	
LR	Fotowiderstand	
BL	Zwerglampe, Kugelform, 3,5 V 0,2 A	
OKO	Optokoppler CNY 17 o.ä.	
R	Relais 50 Ohm 2 Umschaltkontakte	
SU, SO	Mikroschalter	
M	Miniatur-Gleichstrommotor mit Getriebe	

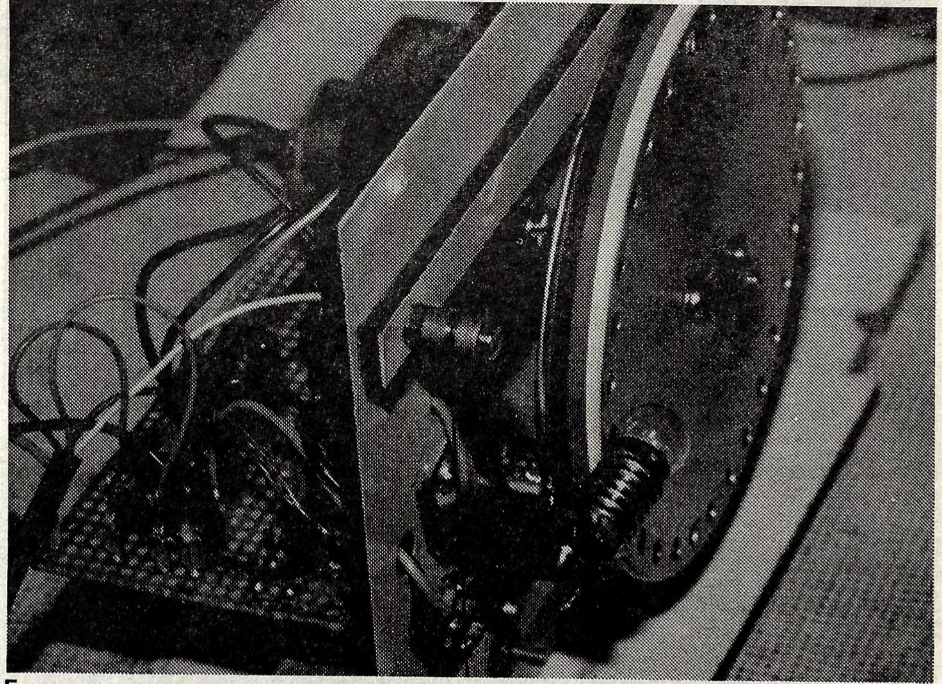
des Mittelpunktes, ist der Zirkel. Aus starkem Zeichenpapier oder -karton macht man sich eine Scheibe, die genau in den Cremedosendeckel paßt und als Bohrschablone dient. Der Zirkeleinstich ist dann schon der Mittelpunkt. Die Lichtschrankenlöcher müssen einen Abstand von 12 Grad haben und sollten auf einem möglichst großen Durchmesser liegen. Wählt man 4 mm Durchmesser für die Löcher, so ergibt sich ein Kreisdurchmesser von 80 mm. Der Mittelpunkt wird genau im Durchmesser der Getriebewelle gebohrt (3 mm). Somit ist erst einmal sichergestellt, daß die Seiltrommel keinen Höhenschlag hat.
Damit die Scheibe auch senkrecht zur Welle befestigt werden kann, braucht sie eine Nabe. Die macht der drehbanklose Bastler aus einem Holzklotz (etwa 25x25x25 mm) in den genau senkrecht zu einer Fläche das Wellenloch gebohrt wird. Das ist freihändig nicht zu schaffen, dazu ist schon ein Bohrständer nötig. Das gebohrte Loch muß genau den Wellendurchmesser haben, damit der Klotz sich stramm auf die Welle schieben läßt und nicht wackelt.

EINFACHSTE HILFSMITTEL SIND GEEIGNET

Die Seiltrommel und der Klotz werden auf die Welle gesteckt (besser vorerst auf das abgeschnittene Stück) und dann zwei Löcher für die Schrauben gebohrt, die Klotz und Scheibe zusammenhalten. Nun fehlt noch ein „Mitnehmer", der eine drehfeste Verbindung zwischen Welle und Seiltrommel herstellt. Den machen wir aus einer „Dosenklemme" von der Elektroinstallation und einen Streifen Messingblech (in Bild 2 zu sehen). Der Blechstreifen bekommt ein genaues Loch für die Welle, wird auf diese gesteckt, die Klemme davorgesetzt und festgeschraubt, dann wird beides verlötet. Nun kommen noch zwei Löcher für die Schrauben in den Blechstreifen, so daß er unter die Schrauben gelegt werden kann, die die Seilscheibe am Klotz halten. Das ist etwas umständlich zu beschreiben, aber das Foto ist dann vielleicht besser zu durchschauen.
Hoffentlich „läuft" Ihre Seiltrommel nachher schön, ohne Höhen- und Seitenschlag!
Den aufzuwickelnden Meß-Faden stört ein Seitenschlag von 1 – 2 mm kaum, doch für eine Reflexlichtschranke stellt er sich als zu groß heraus. Daher wurde das Muster mit einer Loch-Lichtschranke gebaut.

DER SCHWIMMER

Kork ist dafür das beste Material. Ein Flaschenkorken allein ist zu leicht, die richtige Größe hat ein Kork für Gärballons von etwa 40 bis 50 mm Durchmesser (Bild 7). Um den Mikroschalter zu betätigen, muß er an der Unterseite noch etwas Ballast erhalten, das ist auszuprobieren. Er muß auch in die Flüssigkeit eintauchen, ohne umzukippen, wofür der Ballast auch sorgt. In Heizöl taucht der Schwimmer tiefer ein, als in Wasser, das ist zu bedenken.



5.
Blick auf den Rollenhebel und den Lauf des Fadens. Die Rolle ist ein Abstandsröhrchen in das eine Rille gefeilt wurde.

DIE ANSCHLÜSSE AN DEN COMPUTER

Wie oben erwähnt, eignen sich die „Joystickports" als Eingänge für die beiden Signale, die der Füllstandmesser abgibt. Diese Eingänge erwarten keine Fremdspannung, ja sie würde sogar stören. Einzig zulässig sind Widerstände, Kontakte oder Halbleiterstrecken, die ohne Fremdstrom zu steuern sind, also etwa normale Optokoppler. Auch nur eine einpolige Verbindung zwischen den inneren Stromkreisen des Füllstandmessers und dem Computer sollte vermieden werden. Daher müssen die Mikroschalter ihre Signale über einen Optokoppler an den Computer geben.
Der Computer erkennt einerseits Signale, die durch einen Widerstand von weniger als 1000 Ohm erzeugt werden, genauso an, wie einen geschlossenen metallischen Kontakt. Andererseits wirkt ein Widerstand von mehr als 3000 Ohm wie ein offener Kontakt. (Die genauen Werte lassen sich leicht mit einem Drehwiderstand am eigenen Computer feststellen.) Daher kann ein Fototransistor oder eine Fotodiode genausogut wie ein Fotowiderstand für die Lichtschranke benutzt werden, aber kein Fotoelement.
Die Apparatur des Füllstandmessers bleibt selbstverständlich außerhalb des Tanks, nur der Lot-Faden muß ins Innere gelangen, wo der Schwimmer daranhängt. Aber noch etwas ist nötig: die obere Endlage des Schwimmers muß mit einem Microschalter festzustellen sein. Der

ANBAU AN DEN HEIZÖLTANK

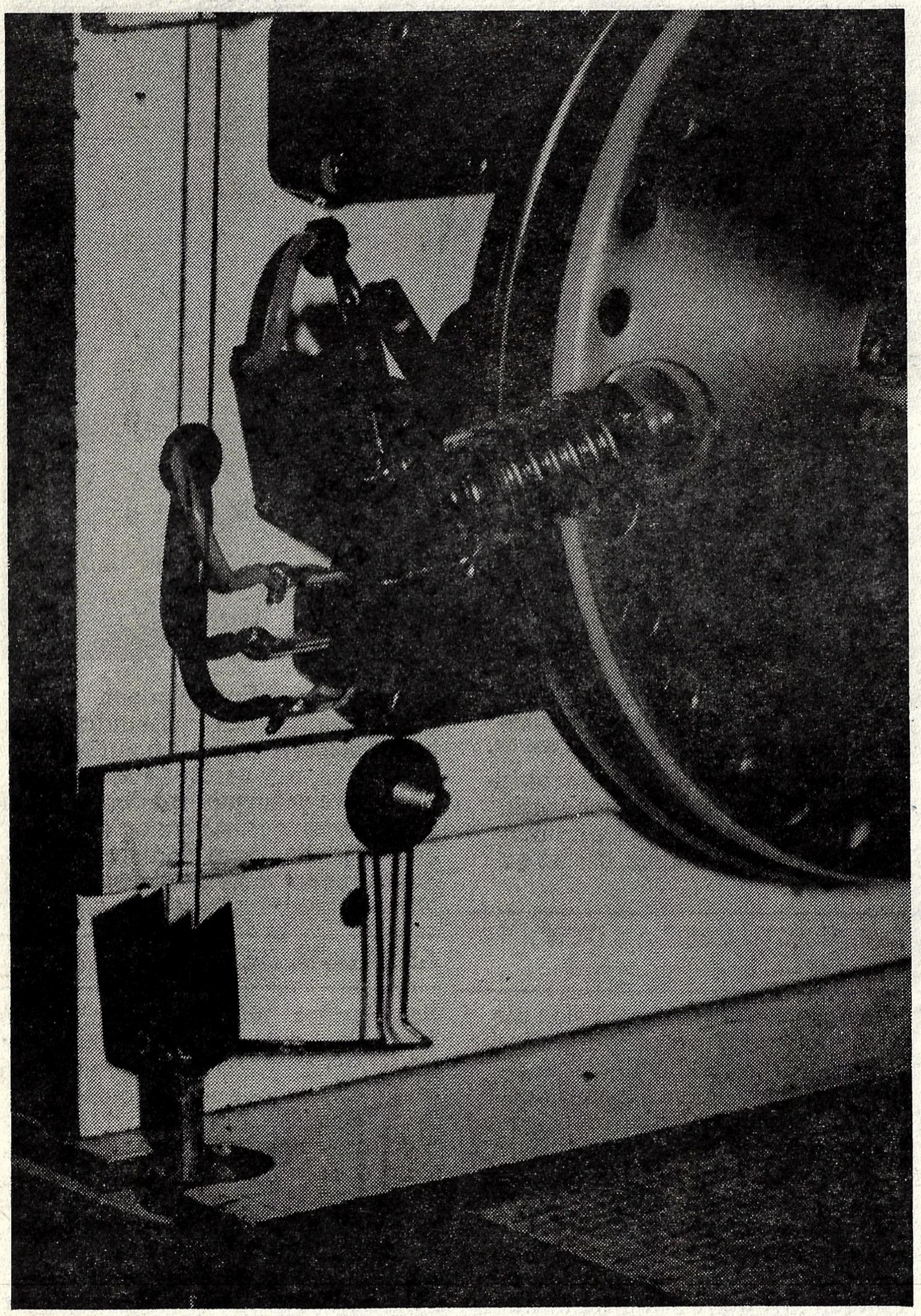
Schwimmer muß also in der oberen Endstellung ein „mechanisches Übertragungsglied" bewegen (Bild 6 und 7), welches außerhalb des Tanks den Schalter betätigt. Das mechanische Übertragungsglied besteht aus einer normalen 4-mm-Steckbuchse und einem Rohrniet von 3 bis 3,5 mm Durchmesser und 30 mm Länge. Die Steckbuchse wird oberhalb des Flüssigkeitsspiegels in die Tankwand eingebaut, der Rohrniet von oben eingeführt, wobei der Nietkopf ihn am Durchfallen hindert. Durch den Niet hindurch führt der Meß-Faden, der innen im Tank am Schwimmer be- »

festigt ist. Wird nun der Schwimmer hochgezogen, so stößt er oben an den Rohrniet und hebt ihn an, dieser wiederum betätigt nun außerhalb des Tanks den Mikroschalter. Wichtig ist, daß der Schwimmer nicht „hart“ anstößt, sondern eine Feder dazwischengeschaltet ist, da der Motor, nachdem er ausgeschaltet ist, noch etwas nachläuft und den Faden etwa 1 cm aufwikkelt. Darum ist die Gabel am unteren Hebel aus Federstahldraht zurechtgebogen.

Wird die ganze Apparatur nicht direkt auf den Tank gebaut (der Meßfaden kann ja – evtl. über Umlenkrollen – noch außerhalb des Tanks weitergeführt werden), so muß aber der Mikroschalter für die obere Endlage des Schwimmers dennoch direkt über der Fadendurchführung angeordnet sein.

Teilebeschaffung:
Einige Teile sind nicht im nächsten Elektronikladen zu haben, für das Mustergerät wurden die folgenden Teile von Conrad Electronik, 8452 Hirschau, bezogen:

1 Getriebemotor-Bausatz 851RS
Bestell-Nr. 24 07 02
1 Relais 6 V, 2xUM
Bestell-Nr. 50 34 36
2 Mikroschalter 1pol. W.
Bestell-Nr. 70 45 12
1 Foto-Widerstand LDR 07
Bestell-Nr. 18 35 55
1 Optokoppler Su 25
Bestell-Nr. 18 41 60



6.
Blick auf den Gabelhebel. Die Gabel besteht aus Federstahldraht, der 2 x L-förmig gebogen ist. Diese federnde Gabel bietet einen weichen Anschlag für den Schwimmer beim Auslaufen des Motors nach dem Ausschalten.

DIE GESCHWINDIGKEIT DES COMPUTERS

Der Computer stellt Ansprüche an die Länge und den zeitlichen Abstand der Signale. So schnell, wie manche Leute behaupten oder vermuten, sind die Apparate in der Praxis gar nicht. Die Megahertz der in Prospekten angegebenen Taktfrequenzen werden zu „einfachen“ Hertz, wenn man die Ausführungsgeschwindigkeit der in BASIC gegebenen Befehle betrachtet. Die Zählimpulsfolge, die von der Lichtschranke der Seiltrommel ausgeht, darf gar nicht so schnell sein. Bei zehn Impulsen je Sekunde kann er sich schon verzählen, d.h., einen oder einige Impulse auslassen. Das muß natürlich berücksichtigt werden. Die Seiltrommel darf also nicht beliebig schnell laufen. Mit dem hier vorgeschlagenen Getriebemotor und 5 Volt Betriebsspannung dreht sich die Seiltrommel in fünf Sekunden einmal. Dementsprechend ist die Zählimpulsfolge 6/s, oder 6 Hz. Das scheint nicht sehr schnell zu sein. Für den Texas-Instruments-Heimcomputer 99/4A darf es kaum mehr sein.
Soll die Füllstandmessung richtig funktionieren, muß der Computer während der Hellzeit der Lichtschranke, also bei jedem Loch in der Seiltrommel, mindestens zweimal das Signal prüfen, ehe er es als „gezählt“ bewertet.

DAS PROGRAMM FÜR DIE FÜLLSTANDMESSUNG

Der TI99/4A hat spezielle (Frage-) Befehle, mit denen die Stellungen des Steuerknüppels und der Zustand des Auslöseknopfes (Trigger) als „Wert“ in das Programm übernommen werden können, nämlich:
CALL JOYST und CALL KEY
Die Bezeichnung „Befehl“ ist eigentlich nicht recht zutreffend, denn das Schlüsselwort stellt die Frage an den Joystick: „Wie ist deine augenblick-

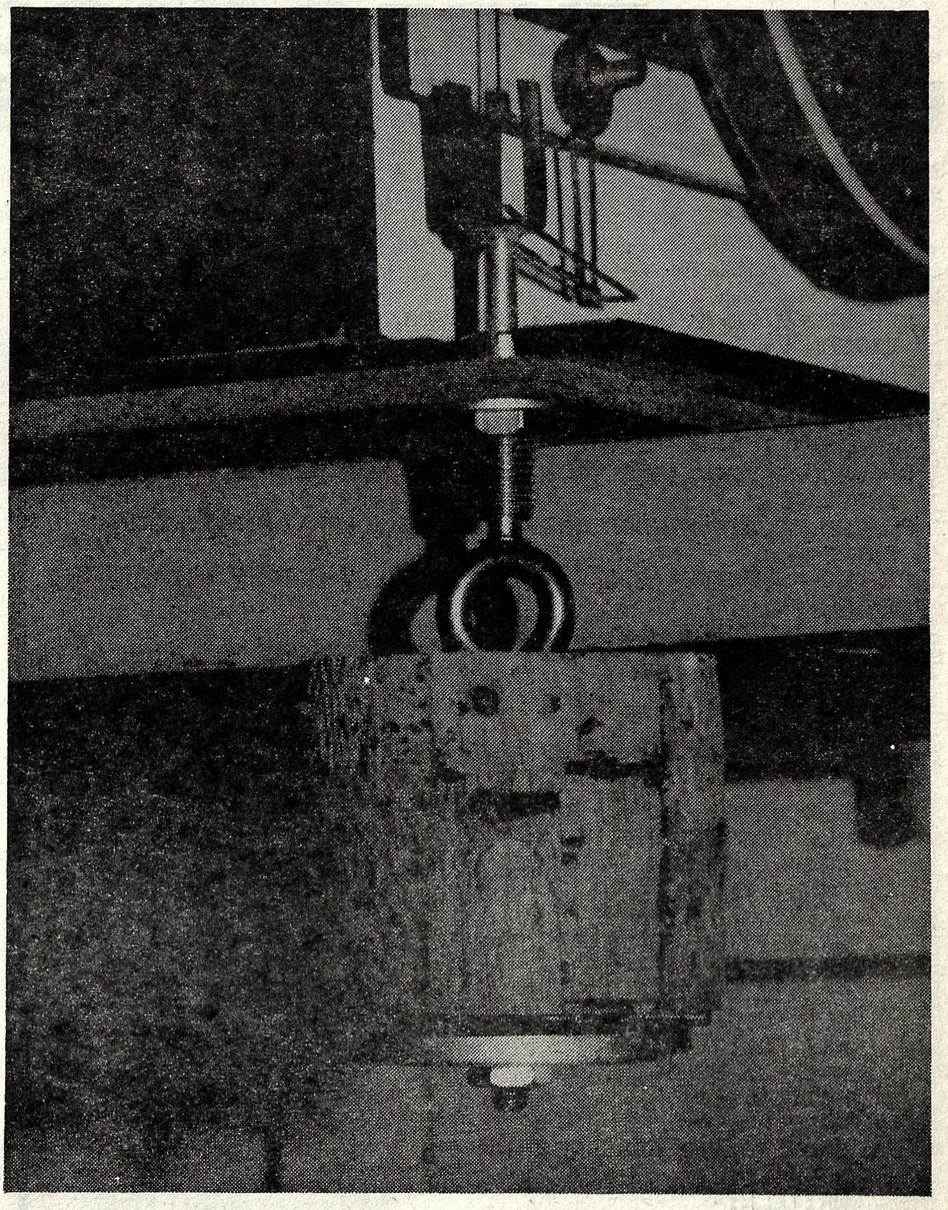
liche Stellung?“ und der antwortet darauf mit Zahlen, d.h., weist zwei Variablen positive oder negative numerische Werte zu.
Immer ist eine Programmzeile mit IF nötig, um die Stellung des Joystick- oder Triggerkontaktes herauszubekommen. Das sieht dann so aus, wie in den Zeilen 140/150.
Dieses Beispiel zeigt zugleich die für den Anwendungsfall sehr wichtige „Warteschleife“. Das heißt, das Programm läuft bei diesen Zeilen nicht weiter, da sie immer wieder wiederholt werden. Erst wenn der Kontakt betätigt ist U=O wird, geht es mit der nächsten Zeile weiter.
Das folgende Programm läuft auf dem TI 99/4A in normalem TI-BASIC:

```
1ØØ REM TEXAS 99/4A-Programm
 fuer Fuellstandmesser
11Ø PRINT "Heizoelvorrat mes
sen"
12Ø PRINT "Bitte Taste druec
ken!"
13Ø A=Ø
14Ø CALL JOYST(1,Z,U)
15Ø IF U=-4 THEN 14Ø
16Ø PRINT "Messung laeuft!"
17Ø CALL JOYST(1,Z,U)
18Ø IF Z=4 THEN 17Ø
19Ø A=A+1
2ØØ CALL JOYST(1,Z,U)
21Ø IF U=-4 THEN 24Ø
22Ø IF Z=Ø THEN 2ØØ
23Ø GOTO 17Ø
24Ø PRINT "Messergebnis ";A;
"cm"
25Ø CALL JOYST(1,Z,U)
26Ø IF U<>-4 THEN 25Ø
27Ø INPUT "Neue Messung: ENT
ER druecken";Q$
28Ø GOTO 12Ø
```

Der Zusammenhang zwischen den Funktionen des Füllstandmessers und den Signalen zeigt die Tabelle:

Signale beim Ablauf:	Z	U
Ruhe	0	–4
Senken........ hell=	4	0
Unten	0	–4
Heben	0	0

Der Joystickanschluß ist an der linken Seite des Geräts. Der Zählausgang des Füllstandmessers kommt an Pin 9 und Pin 7. Der Anschluß der Steuerkontakte über den Optokoppler an Pin 8 (Collector des Optokopplers) und Pin 7 (Emitter des Optokopplers).
Das vorstehende Programm eignet sich erst einmal für „Trockenübungen“: Das Füllstandmeßgerät wird betriebsfertig an den Tischrand gestellt, und die Höhe des Tisches gemessen. Dabei ist allerdings der Abstand von der Unterfläche des Schwimmers zur Tischoberfläche hinzuzurechnen (das kann man ja auch als „Konstante“ programmieren).



7.
Hier sind der Schwimmer, die Durchführung des Fadens durch die Tankwand sowie die Wirkung des „mechanischen Übertragungsgliedes“ und der federnden Gabel genauer zu sehen.

Bis jetzt zählt der Computer lediglich „Höheneinheiten“, beim Mustergerät sind es ganze Zentimeter. Daraus müssen die Öl-Liter errechnet werden. Das ist ja wohl kein allzu großes mathematisches Problem, wenn der Öltank rechteckig ist. Aus Länge und Breite des Tanks ist die Literzahl für die „Höheneinheit“, also Liter je cm, leicht zu berechnen, nämlich: Länge in cm mal Breite in cm, geteilt durch 1000.

Für zylindrische oder kugelförmige Tanks sind die mathematischen Beziehungen etwas schwieriger.
Das Buch von Rudolf Busch, Basic: Mathematik per Computer, Band 5 der Computer Bibliothek im Franzis-Verlag, hilft dann weiter.
Falls der Tankhersteller einen Peilstab zur Füllstandkontrolle mitgeliefert hat (bei Erdtanks üblich), kann man alle Rechnerei umgehen und die Skala des Stabes mit dem cm-Zählwerten des Füllstandmessers in DATA-Zeilen übernehmen.
Der perfektere Computer—Operator wird dann auch den gemessenen Stand zusammen mit dem Datumspeicher und aus den vorhergehenden Messungen den durchschnittlichen Verbrauch automatisch berechnen. Anhand dieser Zahl ist es dann möglich, auch den Tag im Voraus zu berechnen, an dem neues Öl bestellt werden muß.

TURBO PASCAL UND MASCHINENSPRACHE

```
PROGRAM TEST;

  VAR I : INTEGER;
      B : INTEGER;
      C : INTEGER;

  PROCEDURE POKEV(ADRESSE : INTEGER; WERT : INTEGER);
    EXTERNAL;
  PROCEDURE PEEKV(ADRESSE : INTEGER; VAR WERT : INTEGER);
    EXTERNAL;
  PROCEDURE POKE(ADRESSE : INTEGER; WERT : INTEGER);
    EXTERNAL;
  PROCEDURE PEEK(ADRESSE : INTEGER; VAR WERT : INTEGER);
    EXTERNAL;
  PROCEDURE BITSET(ADRESSE : INTEGER; WERT : INTEGER);
    EXTERNAL;
  PROCEDURE BITTST(ADRESSE : INTEGER; VAR WERT : INTEGER);
    EXTERNAL;

BEGIN
  WRITELN;
  WRITELN(" MEMLIB - TESTPROGRAM");
  WRITELN;
  WRITELN(" C BY R.ZIEGLER MOOSBURG");
  WRITELN;
  WRITELN;
    WHILE TRUE DO BEGIN
      WRITE(" MEMORY ADRESSE = ");
      READ(I);
      WRITELN;
      WRITE(" AUSGABEWERT     = ");
      READ(B);
      POKE(I,B);
      PEEK(I,C);
      WRITELN;
      WRITE(" GELESENER WERT = ");
      WRITE(C);
      WRITELN;
      WRITE(" VDP ADRESSE     = ");
      READ(I);
      WRITELN;
      WRITE(" AUSGABEWERT     = ");
      READ(B);
      POKEV(I,B);
      PEEKV(I,C);
      WRITELN;
      WRITE(" GELESENER WERT = ");
      WRITE(C);
      WRITELN;
      WRITE(" CRU ADRESSE     = ");
      READ(I);
      WRITELN;
      WRITE(" AUSGABE 0/1     = ");
      READ(B);
      BITSET(I,B);
      WRITELN;
      WRITE(" CRU ADRESSE     = ");
      READ(I);
      BITTST(I,B);
      WRITELN;
      WRITE(" GELESENER WERT = ");
      WRITE(B);
      WRITELN;
      WRITELN;
    END;
END.
```

Leider haben es die Autoren von Turbo Pasc'99 versäumt, näher auf die Parameterübergabe zwischen den beiden Sprachen einzugehen. Dieses Versäumnis möchte ich hier zum Teil nachholen.
Generell muß zwischen zwei Aufrufen unterschieden werden. Zum ersten, ob ich nur Daten ans Maschinen-Programm übergeben will, oder ob zweitens auch Daten ins Pascal übernommen werden.
Beschreiben wir zunächst Fall Eins. Ein Turbo-Pascal-Programm könnte z.B. so aussehen:
Zulässige Werte für die einzelnen Prozeduren, Angaben in Hex, die aber dezimal eingegeben werden müssen.

	ADRESSE	WERT	
POKEV	0–3FFF	0–FF	VDP WRITE
PEEKV	0–3FFF	0–FF	VDP READ
POKE	0–FFFF	0–FF	MEMORY WRITE
BITSET	0–1FFF	0 / 1	CRU WRITE
BITTST	0–1FFF	0 / 1	CRU READ

Die Routinen BITSET und BITTST beziehen sich auf CRU-Ausgaben. Der Bereich für Adresse liegt zwischen 0 bis 8191 (HEX 0 bis 1FFF). Als Wert wird 0 oder 1 übergeben (BITSET) bzw. 0 oder 1 gelesen (BITTST). Diese Routinen könnten auch durch LDCR R1,8 bzw. STCR R3,8 auf 8 Bit breite Ausgaben erweitert werden.
Die Prozeduren POKEV und PEEKV gestatten den Zugriff von Pascal aus INS VDP Memory (z.B. Adresse = 4200, Wert = 64, gibt ein Zeichen auf dem Bildschirm aus). POKE und PEEK erlauben Modifikationen des Prozessors Memories unter Turbo Pasc'99.
Nach FCTN 9 und CO DSK1.TEST (SA DSK1.XXX nicht vergessen), liegt nun das Testfile vor und muß nur noch, wie gewohnt, assembliert werden. Z.B. in DSK1.TEST*
Wie sieht nun das entsprechende Maschinenprogramm aus?
Generell läßt sich sagen, daß es relocatibel ab >A000 liegen muß, also kein AORG usw. angeben.
Zuerst wird der Interrupt abgeschaltet, dann wird mit AI R0,>FFFX festgelegt, wieviele Parameter übergeben werden.
Einige Beispiele:

1 Parameter	AI	R0,>FFFE
2 Parameter	AI	R0,>FFFC
3 Parameter	AI	R0,>FFFA
4 Parameter	AI	R0,>FFF8

```
* FILENAME : DSK1.MEMLIB*
* Type     : Dis/Fix 80
* Code     : Compressed
* Loader   : Standard
* Loads at : >A000    RORG
* R. ZIEGLER MOOSBURG

       DEF   PEEKV,POKEV
       DEF   PEEK,POKE
       DEF   BITTST,BITSET

AA     BSS   32                 PLATZ FUER REGISTERSATZ

PEEKV  LIMI >0000               INTERRUPT AUS
       AI    R0,>FFFC           2 PARAMETER UEBERGEBEN
       LWPI AA                  REGISTERSATZ LADEN

       MOV   @>8300,R2          POINTER AUF PARAMETER NACH R2
       MOV   *R2+,R0            ADRESSE NACH R0
       SWPB R0                  LSB DER ADRESSE
       MOVB R0,@>8C02           NACH VDPWA
       SWPB R0                  MSB DER ADRESSE
       MOVB R0,@>8C02           NACH VDPWA
       MOV   *R2,R1             POINTER AUF VAR NACH R1
       CLR   R0                 R0 LOESCHEN
       MOVB @>8800,R0           VDPRD NACH R0
       SWPB R0                  INS LSB SCHIEBEN
       MOV   R0,*R1             NACH INHALT R1 (POINTER) LADEN

       LWPI >8300               WORKSPACE AUF >8300 LEGEN
       LIMI >0002               INTERRUPT ZULASSEN
       B     *R11               RUECKSPRUNG INS PASCAL

POKEV  LIMI >0000               INTERRUPT AUS
       AI    R0,>FFFC           2 PARAMETER UEBERGEBEN
       LWPI AA                  REGISTERSATZ LADEN

       MOV   @>8300,R2          POINTER AUF PARAMETER NACH R2
       MOV   *R2+,R0            ADRESSE NACH R0
       ORI   R0,>4000           HOECHSTWERTIGE BITS AUF 01 SETZEN
       SWPB R0                  LSB DER ADRESSE
       MOVB R0,@>8C02           NACH VDPWA
       SWPB R0                  MSB DER ADRESSE
       MOVB R0,@>8C02           NACH VDPWA
       MOV   *R2,R1             AUSGABEWERT NACH R1
       SWPB R1                  INS MSB SCHIEBEN
       MOVB R1,@>8C00           VDPWD

       LWPI >8300               WORKSPACE AUF >8300 LEGEN
       LIMI >0002               INTERRUPT ZULASSEN
       B     *R11               RUECKSPRUNG INS PASCAL

POKE   LIMI >0000
       AI    R0,>FFFC
       LWPI AA

       MOV   @>8300,R2          POINTER AUF UEBERGABEPARAMETER NACH R2
       MOV   *R2+,R1            ADRESSE NACH R1
       MOV   *R2,R3             WERT NACH R3 (LSB)
       SWPB R3                  INS MSB SCHIEBEN
       MOVB R3,*R1              AUSGABE WERT NACH ADRESSE

       LWPI >8300
       LIMI >0002
       B     *R11
```

```
PEEK   LIMI >0000
       AI   R0,>FFFC
       LWPI AA

       MOV  @>8300,R2        POINTER AUF PARAMETER NACH R2
       MOV  *R2+,R1          ADRESSE NACH R1
       MOV  *R2,R3           !! ACHTUNG, POINTER AUF VAR NACH R3
       CLR  R4               R4 LOESCHEN
       MOVB *R1,R4           BYTE LESEN
       SWPB R4               INS LSB GEBEN
       MOV  R4,*R3           R4 NACH INHALT VON R3 KOPIEREN

       LWPI >8300
       LIMI >0002
       B    *R11

BITSET LIMI >0000
       AI   R0,>FFFC
       LWPI AA

       MOV  @>8300,R2
       MOV  *R2+,R12         CRU ADRESSE NACH R12
       MOV  *R2,R1           WERT(0,1) NACH R1
       SWPB R1               INS MSB SCHIEBEN
       LDCR R1,1             AUSGABE 1 BIT

       LWPI >8300
       LIMI >0002
       B    *R11


BITTST LIMI >0000
       AI   R0,>FFFC
       LWPI AA

       MOV  @>8300,R2
       MOV  *R2+,R12         CRU ADRESSE NACH R12
       MOV  *R2,R1           POINTER AUF VAR NACH R1
       CLR  R3               R3 LOESCHEN
       STCR R3,1             1 BIT EINLESEN
       SWPB R3               INS LSB SCHIEBEN
       MOV  R3,*R1           ERGEBNIS NACH INHALT R1(POINTER) KOPIEREN

       LWPI >8300
       LIMI >0002
       B    *R11

       END
```

Pro Integer-Parameter werden 2 Bytes reserviert.
Nach dem Laden des Registersatzes kann nun das eigentliche Maschinenprogramm folgen.
Die Parameterübergabe funktioniert wie nachfolgend beschrieben.
Der Pointer oder Zeiger zu den Parameterdaten liegt in >8300.
Hier steht also die Adresse, ab der die Daten, in der Reihenfolge der Deklaration im Procedure Kopf, liegen.
Beispiel: POKE
>8300 Pointer auf Parameter z.B. >A0E8
>A0E8 Adresse 2 Byte
>A0EA Wert 2 Byte (nur LSB gültig)

Beispiel: PEEK
Hier besteht nun ein wesentlicher Unterschied zu POKE, da hier Daten von Maschinensprache ans Pascal übergeben werden.
>8300 Pointer auf Parameter z.B. >A0E8
>A0E8 Adresse 2 Byte
>A0EA !! Pointer auf VAR Wert: Integer z.B. >A1E0
>A1E0 hierin muß das Ergebnis geladen werden

Diese Beispiele sind noch einmal im nachfolgenden Maschinenprogramm näher erklärt.
Der Rücksprung aus Maschine in Turbo Pasc'99 sieht so aus:

LWPI	>8300	Registersatz auf >8300 legen
LMI	>0002	Interrupt wieder zulassen
B	*R11	Rücksprung ins Turbo Pasc'99

Nach dem Assemblieren des Maschinen-Programms in DSK1.MEMLIB*, kann es zusammen mit dem Hauptprogramm DSK1.TEST*, unter Zuhilfenahme des Linkers LK99, in ein lauffähiges Programm verwandelt werden. *Reiner Ziegler*

```
**********************************************************
*                                                        *
*   EDITOR ASSEMBLER MODUL,UTILITY ROUTINEN ANALYSE     *
*                                                        *
*             19.10.86  Alexander Hulpke                 *
*                                                        *
**********************************************************


2000 A55A DATA >A55A    ID Code

                        XML Vektoren
2002 2128 DATA >2128    Execute MCode
2004 2398 DATA >2398    Lade Tagged File
2006 225A DATA >225A    CIF
2008 FF00 DATA >FF00    Reserviert
BIS  >2020

                        ULTAB DATA AREA
2022 0000 DATA ULTAB    Entry Adress
2024 A000 DATA FSTHI    1.Freie Adresse in HM
2026 FFD7 DATA LSTHI    Letzte freie "  in HM
2028 2676 DATA FSTLOW   1.     "   "  in LM
202A 3F38 DATA LSTLOW   Letzte "   "  in LM
202C 0000 DATA CHKSAV   Checksum
202E 0000 DATA FLGPTR   Pointer zu Flag im PAB
2030 0000 DATA SVGPRT   GPL Return Adress
2032 0000 DATA SAVCRU   CRU Adresse des Peripheriegeraets
2034 0000 DATA SAVENT   Startadresse von DSR oder Subprogram
2036 0000 DATA SAVLEN   Laenge des Namens von DSR oder "
2038 0000 DATA SAVPAB   Pointer zu Namen in PAB
203A 0000 DATA SAVVER   Version Nummer der DSR

203C 0000 BSS >2C       Databuffer

2094 0000 BSS >6        UTILWS
                               Workspaces ueberschneiden sich,da nur
209A 0000 BSS >20       WS#2   R13-15 bei gegenseitigem BLWP Aufruf konstant
                               bleiben muessen !
20BA 0000 BSS >20       WS#3

20DA 0000 BSS >20       WS#4

* Diverse Konstanten

20FA 0064 DATA >0064    DEZ 100 fuer CIF-Routine
20FC 2000 DATA >2000    BIT 2 fuer setze/loesche EQ-BIT in calling Status
20FE 2E   BYTE >2E      ASCII fuer '.'
20FF AA   BYTE >AA      ROM Header Code

* Utility BLWP Vektoren

2100 2094 DATA UTILWS   GPLLNK
2102 21C4 DATA >21C4
2104 2094 DATA UTILWS   XMLLNK
2106 2196 DATA >2196
2108 2094 DATA UTILWS   KSCAN
210A 21DE DATA >21DE
210C 2094 DATA UTILWS   VSBW
210E 21F4 DATA >21F4
2110 2094 DATA UTILWS   VMBW
2112 2200 DATA >2200
2114 2094 DATA UTILWS   VSBR
```

E/A-ROUTINEN DOKUMENTIERT

In den folgenden Ausgaben der TI-REVUE werden die Editor/Assembler Unterroutinen dokumentiert aufgelistet. Diese Auflistung ermöglicht einerseits das gezielte Arbeiten mit dem vom E/A-Modul verwendeten RAMs und andererseits die Verwendung von, schon im Modul enthaltenen, Unterprogrammen. Das besondere Interesse wird mit Sicherheit dem Loader gehören, der im E/A-Modul vollkommen in Maschinensprache geschrieben ist.

```
2116 22ØE DATA >22ØE
2118 2Ø94 DATA UTILWS   VMBR
211A 221A DATA >221A
211C 2Ø94 DATA UTILWS   VWTR
211E 2228 DATA >2228
212Ø 2Ø9A DATA WS#2     DSRLNK
2122 22B2 DATA >22B2
2124 2ØDA DATA WS#4     LOADER
2126 23BA DATA >23BA

* XML >22 : Programmsuche

2128 C8ØB MOV  R11,@SVGPRT  Save R11
212A 2Ø3Ø
212C DØ6Ø MOVB @>8349,R1
212E 8349
213Ø 2Ø6Ø COC  @>2ØFC,R1 BIT 3 gesetzt ?
2132 2ØFC
2134 132A JEQ  >218A       Ja, loeschen und zurueck
2136 CØ2Ø MOV  @>835Ø,RØ
2138 835Ø
213A 1311 JEQ  >215E
213C Ø6AØ BL   @>2646 offene REF'S ?
213E 2646
214Ø 1Ø1E JMP  >217E  Ja,weiter mit ERROR schreiben, Set COND-BIT

2142 Ø281 CI   R1,>3F38 Tabelle durch ?
2144 3F38
2146 1319 JEQ  >217A       Ja, nicht gefunden
2148 CØØ1 MOV  R1,RØ
214A Ø2Ø2 LI   R2,FAC
214C 834A
214E 8CBØ C    *RØ+,*R2+ Test,ob Name in FAC=gespeichertem Namen
215Ø 1611 JNE  >2174       ungleich,naechster Name
2152 8CBØ C    *RØ+,*R2+
2154 16ØF JNE  >2174
2156 8CBØ C    *RØ+,*R2+
2158 16ØD JNE  >2174
215A C81Ø MOV  *RØ,@ULTAB Namen sind gleich, Einsprungsadresse in ULTAB
215C 2Ø22
215E Ø2EØ LWPI WS#3 Hilfsworkspace laden
216Ø 2ØBA
2162 CØ2Ø MOV  @ULTAB,RØ Startadresse holen
2164 2Ø22
2166 13Ø9 JEQ  >217A       =Ø?
2168 Ø69Ø BL   *RØ         Exec
216A Ø2EØ LWPI GPLWS       Rueckkehr
216C 83EØ
216E C2EØ MOV  @SVGPRT,R11 Ruecksprung
217Ø 2Ø3Ø
2172 Ø45B RT

2174 Ø221 AI   R1,>ØØØ8    Naechster Eintrag
2176 ØØØ8
2178 1ØE4 JMP  >2142       Schleife
217A Ø2ØØ LI   RØ,>ØFØØ    'PROGRAM NOT FOUND' ERROR
217C ØFØØ
217E D8ØØ MOVB RØ,@>8322 ERROR schreiben
218Ø 8322
2182 Ø2EØ LWPI GPLWS      Weiter mit
2184 83EØ                 GPL : Set COND.-BIT
2186 Ø46Ø B    @>ØØCE
2188 ØØCE
```

```
218A 582Ø SZCB @>2ØFC,@>8349 BIT 3 loeschen
218C 2ØFC
218E 8349
219Ø Ø2EØ LWPI UTILWS
2192 2Ø94
2194 Ø38Ø RTWP                 und zurueck
```

* XMLLNK ROUTINE

```
2196 C83E MOV  *R14+,@>83E2 DATA in R1 GPLWS
2198 83E2
219A Ø2EØ LWPI GPLWS
219C 83EØ
219E C8ØB MOV  R11,@>2ØAA   in R11 UTILWS (quasi SAVRTN)
21AØ 2ØAA
21A2 CØ81 MOV  R1,R2        DATA Backup
21A4 Ø281 CI   R1,>8ØØØ     Wert groesser >8ØØØ ?
21A6 8ØØØ
21A8 1BØ7 JH   >21B8        Ja,Exec
21AA Ø9C1 SRL  R1,12        sonst Zahl aufbereiten
21AC ØA11 SLA  R1,1
21AE ØA42 SLA  R2,4
21BØ Ø9B2 SRL  R2,11
21B2 AØA1 A    @>ØCFA(R1),R2 Tabellensuche 1. Nybble
21B4 ØCFA
21B6 CØ92 MOV  *R2,R2     2.Nybble
21B8 Ø692 BL   *R2        Routine Ausfuehren
21BA Ø2EØ LWPI UTILWS
21BC 2Ø94
21BE C8ØB MOV  R11,@>83F6  in R11 GPLWS
21CØ 83F6
21C2 Ø38Ø RTWP              und zurueck
```

* GPLLNK ROUTINE

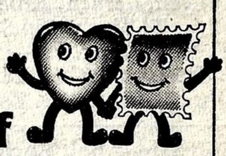
```
21C4 DØ6Ø MOVB @>8373,R1    GPL Substack in
21C6 8373
21C8 Ø981 SRL  R1,8         R1
21CA C87E MOV  *R14+,@>83Ø4(R1) DATA auf Substack
21CC 83Ø4
21CE F82Ø SOCB @>2ØFC,@>8349 BIT 2 setzen
21DØ 2ØFC
21D2 8349
21D4 Ø2EØ LWPI GPLWS
21D6 83EØ
21D8 C2EØ MOV  @SVGPRT,R11   Adresse lesen
21DA 2Ø3Ø
21DC Ø45B RT  Exec und Ruecksprung
```

* KSCAN ROUTINE

```
21DE Ø2EØ LWPI GPLWS
21EØ 83EØ
21E2 C8ØB MOV  R11,@>2ØAA   Save RTN
21E4 2ØAA
21E6 Ø6AØ BL   @SCAN        Scan KBD
21E8 ØØØE
21EA Ø2EØ LWPI UTILWS
21EC 2Ø94
21EE C8ØB MOV  R11,@>83F6   RTN zurueck
21FØ 83F6
21F2 Ø38Ø RTWP
```

```
* VSBW

21F4 06A0 BL     @>223A Adresse Aufbereiten
21F6 223A
21F8 D82D MOVB @>0002(R13),@VDPWD Byte schreiben
21FA 0002
21FC 8C00
21FE 0380 RTWP    zurueck

* VMBW

2200 06A0 BL     @>223A Adresse Aufbereiten
2202 223A
2204 D831 MOVB *R1+,@VDPWD  Byte Schreiben
2206 8C00
2208 0602 DEC   R2
220A 16FC JNE   >2204   Schreibschleife
220C 0380 RTWP  zurueck

* VSBR

220E 06A0 BL     @>2240 Adresse Aufbereiten
2210 2240
2212 DB60 MOVB @VDPRD,@>0002(R13) Lese Byte
2214 8800
2216 0002
2218 0380 RTWP

* VMBR

221A 06A0 BL     @>2240 Adresse Aufbereiten
221C 2240
221E DC60 MOVB @VDPRD,*R1+ Lese Byte
2220 8800
2222 0602 DEC   R2
2224 16FC JNE   >221E  Leseschleife
2226 0380 RTWP

* VWTR

2228 C05D MOV   *R13,R1     R0 holen
222A D82D MOVB @>0001(R13),@VDPWA  Wert schreiben
222C 0001
222E 8C02
2230 0261 ORI   R1,>8000    Registernummer
2232 8000
2234 D801 MOVB R1,@VDPWA    schreiben
2236 8C02
2238 0380 RTWP

* Adressaufbereitung fuer VDP Write

223A 0201 LI    R1,>4000
223C 4000
223E 1001 JMP   >2242

* Adressaufbereitung fuer VDP Read

2240 04C1 CLR   R1

* Gemeinsam weiter

2242 C09D MOV   *R13,R2   Adresse lesen
```

```
2244 D820 MOVB @>2099,@VDPWA und in VDP schreiben
2246 2099
2248 8C02
224A E081 SOC  R1,R2
224C D802 MOVB R2,@VDPWA
224E 8C02
2250 C06D MOV  @>0002(R13),R1    R1&
2252 0002
2254 C0AD MOV  @>0004(R13),R2    R2 des aufrufenden Workspaces uebertragen
2256 0004
2258 045B RT   weiter

* XML >26 : CIF

225A 0204 LI   R4,FAC
225C 834A
225E C014 MOV  *R4,R0  Lese Integer
2260 C184 MOV  R4,R6   Erste beiden Worte
2262 04F6 CLR  *R6+    in
2264 04F6 CLR  *R6+    FAC loeschen
2266 C140 MOV  R0,R5   wenn Zahl=0 dann
2268 1323 JEQ  >22B0   Return
226A 0740 ABS  R0
226C 0203 LI   R3,>0040  Exponent 0
226E 0040
2270 04F6 CLR  *R6+  Loesche Rest von FAC
2272 04D6 CLR  *R6
2274 0280 CI   R0,>0064  Zahl < 100
2276 0064
2278 1A13 JL   >22A0
227A 0280 CI   R0,>2710  Zahl < 10000
227C 2710
227E 1A08 JL   >2290
2280 0583 INC  R3      Exponent erhoehen
2282 C040 MOV  R0,R1   Verschieben fuer DIV
2284 04C0 CLR  R0
2286 3C20 DIV  @>20FA,R0  Zahl/100
2288 20FA
228A D920 MOVB @>83E3,@>0003(R4) Low Byte Ergebnis in Byte 4 FAC
228C 83E3                         (10000er)
228E 0003
2290 0583 INC  R3  Zehnerpotenz erhoehen
2292 C040 MOV  R0,R1  DIV wie oben
2294 04C0 CLR  R0
2296 3C20 DIV  @>20FA,R0
2298 20FA
229A D920 MOVB @>83E3,@>0002(R4)  100 er
229C 83E3
229E 0002
22A0 D920 MOVB @>83E1,@>0001(R4)  1 er
22A2 83E1
22A4 0001
22A6 D520 MOVB @>83E7,*R4   Exponent in FAC
22A8 83E7
22AA 0545 INV  R5      Zahl negativ ?
22AC 1101 JLT  >22B0
22AE 0514 NEG  *R4     Ja, Vorzeichen setzen
22B0 045B RT             Rueckkehr in XML
```

Zeig beim Porto
Herz & Verstand:

Kauf
Wohlfahrts-
briefmarken
Hilfe, die ihr Ziel erreicht.
Erhältlich bis Ende März bei der Post
ganzjährig bei den Wohlfahrtsverbänden

TI tit REVUE

Das Magazin für TI 99-4A

Nr.6/87

Mini-Modul-Expander für GRAM-MODUL

Dialog

Tips & Tricks

Clubs

Andere Baud-Rate für RS 232

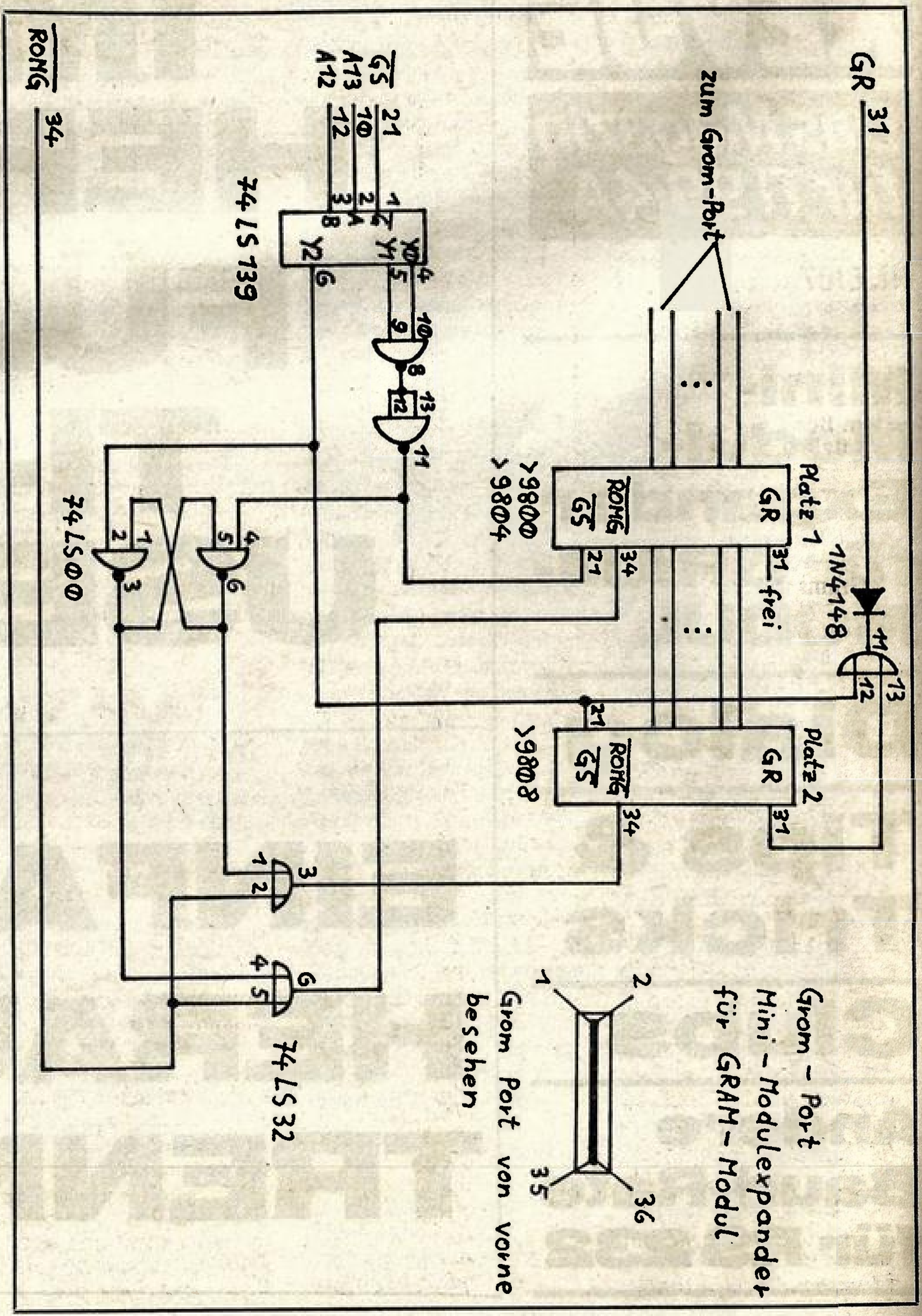
DAS HEFT IM HEFT FÜR TI-USER

EINFACH HERAUS-TRENNEN

MINI-MODUL EXPANDER FÜR GRAM-MODUL

Dieser Expander ist speziell für das 40 KByte Gram-Modul gedacht, da mit seiner Hilfe Module sehr leicht ausgelesen oder parallel zum Gram-Modul betrieben werden können.
Einfach haben es Besitzer von Navarone Modulexpandern.
Sie müssen nur die Änderungen verdrahten und sparen sich die komplette Busverkabelung. Alle Signale können, vom Grom-Port kommend, parallel verbunden werden, außer GR(GROM READY), GS (GROM SELECT) und ROMG (ROM ENABLE).
Die zuletzt genannten Signale werden von der TTL-Logik generiert und mit den entsprechenden Pins der Modulplätze 1 und 2 verbunden.
Im ersten Platz sind zwei Gromadressen gültig, nämlich die vom Gram-Modul geforderten >9800 für RAM und >9804 für Editor/Assembler. Platz 2 bekommt demnach >9808 (wichtig als Grombasis zum Auslesen der Module) zugewiesen.
Wichtig auch zu wissen, daß in Platz 1 Probleme auftreten können, wenn Originalmodule eingesteckt werden. Dies wird durch das fehlende GR-Signal verursacht, mit dessen Hilfe Module mit der Konsole synchronisiert werden. Nachbauten besitzen dieses Signal nicht, da sie wesentlich schneller sind. Platz 1 sollte somit dem 40 KByte Gram-Modul vorbehalten bleiben.
Die vorgestellte Erweiterung stellt eine relativ preiswerte Lösung dar, um Originalmodule gleichzeitig in Verbindung mit von Diskette geladenen zu verwenden.
Ein weiterer Vorteil liegt in der Schonung des Grom-Ports. Ein Punkt, der nicht unterschätzt werden sollte.
Ein wichtiger Hinweis sei mir noch gestattet.
Wurde z.B. das Minimemory ins Gram-Modul geladen, welches diverse Unterprogramme im Basic zur Verfügung stellt (Save Minimem usw.) und wird Extended Basic in Platz 2 gesteckt, so werden beim Anwählen von Ex-Basic auch die speziellen Minimemory-Befehle verfügbar.

Reiner Ziegler

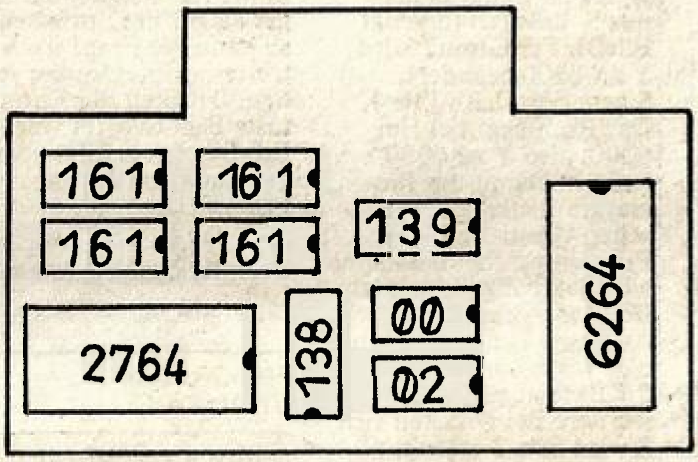


DER BAU

SELBSTBAU 40 KBYTE GRAM-MODUL + 8 KBYTE IMBEREICH HEX 6000 FÜR DEN TI99/4A

Voraussetzungen:
Heiner Martin Grom-Platine Typ 2 + alle Teile
6 Stück 6264 8KByte RAM
1 Stück 2764 8KByte Eprom mit Inhalt Editor/Assembler

An der von H. Martin veröffentlichten Platine Typ 2 (Heft 5/85) lassen sich die notwendigen Hardware-Änderungen relativ einfach durchführen. Sinnvollerweise sollte man die Unterbrechungen auf der Platine vor dem Bestücken und die zusätzlichen Verbindungen nach dem Bestücken vornehmen. Nach den im nachfolgenden Teil beschriebenen Änderungen werden nun die maximal 5 Grams (6264) verdrahtet.
Alle ICs können Huckepack übereinander gesteckt und parallel verlötet werden, ausgenommen Pin 20. Dieser Pin wird entweder verkürzt oder etwas hochgebogen (bei jedem der 5 Chips). Nun werden die 5 RAMs über das bereits eingelötete Eprom (Inhalt Editor/Assembler) gesteckt.
Pin 26 und 27 des untersten RAMs (alle RAM Pin 26 und 27 sind ebenfalls parallel verdrahtet) werden hochgebogen und gesondert verbunden. Alle anderen Pins (außer Pin 20 der 5 Chips) werden parallel verlötet. Danach verlöten Sie die fünf Pin 20 der RAMs mit dem IC 74LS 138 Pin 7, 9, 10, 11 und IC 74LS139 Pin 4 (welcher mit wem ist egal).
Als letztes wird RAM Pin 26 mit IC 74LS00 Pin 11 verbunden und RAM Pin 27 mit 74LS02 Pin 6. Somit kann der Umbau als abgeschlossen betrachtet werden.
Die im folgenden beschriebene Software wurde wegen ihrer Länge nicht komplett aufgelistet und kann über den Verlag geordert werden.



Leitungen, die zu trennen sind:

Die benannten Pins dürfen weder auf der Löt- noch auf der Bestückungsseite angeschlossen sein.

74LS139	Pin 1,2,3,11
74LS 02	Pin 8,9,10,11, 12,13
74LS 00	Pin 11,12,13
74LS138	Pin 11,12
2764	Pin 20,22
6264	Pin 27

Leitungen, die zu verbinden sind:

74LS139	Pin 1 auf Masse Pin 2 mit 74LS 138 Pin 12 Pin 3 mit 6264 Pin 8 Pin 13 mit 74LS02 Pin 8 Pin 15 mit 74LS 02 Pin 9
74LS 02	Pin 10 mit 74LS02 Pin 11 und Pin 12 Pin 13 mit 74LS138 Pin 4
74LS 00	Pin 12 mit 74LS00 Pin 13 Pin 13 mit 74LS139 Pin 3
6264	Pin 26 mit +5V Pin 27 mit 74LS02 Pin 6
2764	Pin 20 mit 74LS139 Pin 6 Pin 22 mit 74LS00 Pin 3

Das notwendige Programm-Paket für das Gram-Modul besteht im wesentlichen aus 4 Hauptprogrammen mit folgenden Funktionen:
DISBUG
zum Auslesen von Modulen
LADER
zum Einlesen von Modulen
GRAMSAVE
zum Abspeichern von Basic-Programmen ins Gram
GPLMOVE
zum Laden eines GPL-Programms ins Gram.

Notwendige Konfiguration:

Editor/Assembler Modul
Disketten-Laufwerk
32 K Speichererweiterung

PROGRAMM DISBUG

Aufruf und Ablauf des Programms:
Nach dem Einstecken des Moduls Editor/Assembler in die Konsole und dem Einschalten von Box und Konsole erfolgt nach dem Erscheinen von
1 FOR TI BASIC
2 FOR EDITOR/ ASSEMBLER
3 FOR REVIEW MODULE LIBRARY
und dem Anwählen von 2Editor/Assembler die nächste Kommandofolge
3 LOAD UND RUN
* LOAD UND RUN*
FILE NAME?
DSK1.DISBUG
FILE NAME? <ENTER>
PROGRAM NAME?
DISBUG
Nachfolgend die Kommandos, die bis auf SLDYZ dem TI-Debugger entsprechen:

A	Load Memory with ASCII
B	Breakpoint Set/Clear
C	CRU Inspect/Change
D	Disassemble Memory
E	Execute
F	Find Word or Byte
G	GROM Base change
H	Hex Arithmetic
I	Inspect Screen Location
K	Find Data Not Equal
L	Change Drive 1 or 2
M	Memory Inspect/ Change
N	Move Block
P	Compare Memory Block
Q	Quit Debugger
R	Inspect/Change WP, PC, & SR
S	Save Memory
T	Trade Screen
U	Toggle Basic offset/ on/off
V	VDP Base change
W	Inspect/Change Register
X	Change Bias
Y	Base Adress
Z	Length File
>	Hex to Decimal convert
.	Decimal to Hex convert

Nach der Meldung des Programms mit
TI 99/4A DISBUG V2.3
erfolgt der Befehl
M 0,FFFF
Jetzt kann das Modul Editor/Assembler gegen das auszulesende Modul ausgetauscht werden.

Achtung:
Pin 1 (RESET) vom auszulesenden Modul mit Tesafilm überkleben (Pin 1 = der Pin, wenn

man in den Modulschacht sieht, links unten).
Mit FCTN X wird das Auflisten abgebrochen.

G Grombasis (beim Start HEX 9800) liegt beim auszulesenden Modul bei HEX 9800, bei eingestecktem Editor/Assembler liegt bei HEX 9800 das oder die Grams und bei Hex 9804 das Modul Editor/Assembler

L schaltet zwischen Laufwerk 1 und 2 um (beim Start 1)

M gestattet das Ändern von Modulnamen usw., aber erst nach dem Wiedereinlesen in das RAM des Gram-Moduls und erneutem Start von Disbug

Y Basis Adresse für Groms oder Roms

GROM 0 >0000 INTERN
GROM 1 >2000 INTERN
GROM 2 >4000 INTERN
GROM 3 >6000 z.B. TI-Writer od. Hustle
GROM 4 >8000
GROM 5 >A000
GROM 6 >C000
GROM 7 >E000
ROM >0000 INTERN
>4000 DSR-ROMS mit CXXXX einschalten
>6000 ROM-Module
RAM alle Adressen gültig

Z Länge des Datenfiles
1800 bei internen Groms und den meisten Modulen
2000 nötig bei speziellen Modulen Testder maximalen Gromlängen z.B. mit M G6000, 8000 bei Grom 3, zur Sicherheit kann immer 2000 angegeben werden.
0000 für RAM oder ROM Adressen, die Filelänge ist immer 8 KByte

S speichert den Inhalt des durch Y,Z und G festgelegten Memories auf das angewählte Laufwerk ab. Die File-Namen werden automatisch generiert.

DSKX.FILE0
DSKX.FILE1
DSKX.FILE2
usw.

Q schließt das Auslesen ab.

Beispiel für ein gedachtes Modul mit 2 Groms und einem Rom (HEX 6000)

Für das erste Grom (Hex 6000) stimmen alle Werte in Y und Z, also nur S drücken (erzeugt File0). Für Grom2 wird Y zu 8000 geändert, S generiert dann File 1. Das Rom liegt bei Hex 6000, also Y zu 6000 ändern. Damit das ProgrammLader später zwischen Grom- und Rom-Files unterscheiden kann, wird jetzt Z zu 0 gesetzt. Bei Ram- oder Rom-Adressen kann somit nur eine feste Länge von 8 KByte abgespeichert werden. Bei Drücken von S wird File 2 erzeugt (Rom-File). Der Lader erkennt dies später automatisch und lädt das File entsprechend.
Mit Hilfe des Disk Managers können die Files nun umbenannt werden.
Z.B.:FILE0 IN GAME1
FILE1 IN GAME2
FILE2 IN GAME3

Anschließend wird in TI-Basic, mit eingestecktem Editor/Assembler, das Programm Lader eingelesen.

PROGRAMM LADER

Aufruf über TI-Basic mit Editor/Assembler Modul

OLD DSK1.LADER

Folgende Funktionen werden vom Lader zur Verfügung gestellt:

– FILE LADEN
Hierbei werden nur einzelne Files ins Ram geladen, sinnvoll bei reinen Rom-Modulen oder Modulen mit nur einem Grom.

– MODUL LADEN
Hier können Module, die aus zwei und mehr Groms bestehen, mit der Angabe eines einzigen Filenamens gestartet werden.

– EDIT FILE
Dieser Editor erzeugt ein DIS/VAR 80 File mit dem gewünschten Filenamen des Moduls (wird am Anfang abgefragt). Danach müssen die maximal 6 Einzelfilenamen eingegeben werden (5 Groms und 1 Rom).
Z.B.:DSK1.GAMES3
DSK1.GAMES1
DSK1.GAMES2
Sollte die Eingabe weniger als 6 Files umfassen, so kann die Frage nach weiteren Files immer mit dem Drücken der Entertaste beantwortet werden. Das Starten erfolgt über die Funktion Modul laden.

– QUIT
Beendet das Programm Lader

PROGRAMM GRAMSAVE

Aufruf über TI-Basic mit Editor/Assembler Modul

OLD DSK1.GRAMSAVE

Dieses Programm gestattet das Verschieben von Basic-Programmen in das Gram. Diese Option beschränkt sich auf TI-Basic-Programme, ohne Verwendung von CALL LINK, CALL LOAD usw. Im Command-Modus können diese, durch den Editor/Assembler zur Verfügung gestellten, Routinen verwendet werden. Probleme traten bisher nur bei Verwendung von Datas auf. Sollten Sie den maximalen Gramausbau (5 Stück) besitzen, so besteht die Möglichkeit, verschieden oder gleichlange Basic-Programme abzuspeichern.

Beispiele:
5 Programme mit je 8 KByte (maximum)
3 mit je 8 KByte und 1 mit 16 KByte
1 mit 8 KByte und 2 mit 16 KByte
usw.

Insgesamt stehen somit 40 KByte für Basicprogramme zur Verfügung

TABELLE 1

–1 –8192	GRAM 7 8 KBYTE
–8193 –16384	GRAM 6 8 KBYTE
–16385 –24576	GRAM5 8KBYTE
–24577 –32768	GRAM4 8 KBYTE
32767 24576	GRAM3 8 KBYTE

Nach dem Starten des Programms Gramsave, muß das eigene Programm geladen werden. Danach erfolgt das Abspeichern ins Gram mit

CALL LINK("SAVE", "NAME",START, ENDE)

Nach FCTN = kann das Programm über das Hauptmenü angewählt werden. Durch Aktivieren von Review Module Library (entsprechende Zahl drücken) kann nun zwischen den geladenen Programmen und dem Modul Editor/Assembler umgeschaltet werden.

Wichtig!
Als Startadresse kommen laut Tabelle 1 nur 5 Werte in Frage, da der TI 99/4A nur hier die Programme erkennt. Die Eingabe des Start-Wertes erfolgt dezimal, mögliche Werte sind also:

24576 (HEX 6000)
–32768 (HEX 8000)
–24576 (HEX A000)
–16384 (HEX C000)
–8192 (HEX E000)

Bei HEX 6000 muß ein Programm liegen, sonst funktioniert der Aufruf über das Hauptmenü nicht.

Die höchste Startadresse wird durch die Anzahl der Grams festgelegt.

FFFF	
>FFFF >E000	Startadresse 5
>DFFF >C000	Startadresse 4
>BFFF >A000	Startadresse 3
>9FFF >8000	Startadresse 2
>7FFF >6000	Startadresse 1

Der Ende-Wert (Eingabe ebenfalls dezimal) muß nur größer als die Basic-Programmlänge sein (Fehlerausgabe falls nicht). Es dürfte also keinen Sinn haben, ein 16 KByte-Programm, beginnend bei –8192 (HEX E000), zu laden, sondern nur bei –16384 (HEX C000) oder tiefer bei –24576 (HEX A000). Die Programme können auch nur, von der Basicadresse aus, aufwärts geladen werden.

ANHANG:

Inhalte einiger Module

MODUL NAME	INHALT
ADVENTURE	G3
ADVERTISER	G3
ALPINER	G3,G4,G5,G6,R0
CARWARS	G3
DIAGNOSTICS	G3
DISK-MANAGER2	G3,G4
EDITOR/ASSEMBLER	G3
HUSTLE	G3
HUNT THE WUMPUS	G3
MINI-MEMORY	G3,R0
MULTIPLAN	G3,G4,G5,G6,G7
MUSIC-MAKER	G3,G4,G5
OTHELLO	G3,R0
PARSEC	G3,G4,G5,R0
SCHACHMEISTER	G3,G4,G5,G6,R0
SPEECH-EDITOR	G3
TERMINAL EMULATOR II	G3,G4,G5,G6,R0
TI-INVADER	G3,R0
TI-WRITER	G3
TUNNELS OF DOOM	G3,G4,G5,G6,G7

Die Tabelle zeigt die vorhandenen Rams im Gram-Modul. Module, die mit Bankumschaltung im Rambereich HEX 6000 (Atari, Extended Basic usw.) arbeiten, funktionieren nicht.

Die Länge des Namens kann maximal 10 Zeichen betragen, was aber ausreichend sein dürfte.

Beispiel:
Basicprogramm mit 16 KByte Länge

CALL LINK("SAVE", "DEMO1", 24576,–24577)

Beispiel:
Basicprogramm mit 20 KByte Länge

CALL LINK("SAVE", "DEMO2",–24576,–1)

PROGRAMM GPLMOVE

Aufruf über TI-Basic mit Editor/Assembler Modul

OLD DSK1.GPLMOVE

Nachdem mit dem auf dem Markt befindlichen GPL-Assemblern ein maximal 8 KByte großes GPL-Prog ramm geschrieben wurde, wird dieses, nach Angabe des Filenamens, automatisch ins Gram geladen (HEX 6000 = Gram 3). Nach FCTN = kann das GPL-Programm über das Hauptmenü angewählt werden.
Ich hoffe, mit diesem Programmpaket die gängigsten Gram-Anwendungen abgedeckt zu haben. Sollten Sie trotzdem noch spezielle Wünsche, Fragen oder Anregungen haben, so bitte ich um eine möglichst detaillierte Beschreibung.

Reiner Ziegler

Ich war durstig und ihr...

Helfen ohne Besserwisserei. Mit Partnern, die Vertrauen verdienen. Die wissen, wo ein Brunnen sinnvoller ist als ein Staudamm, die Schöpfkelle wertvoller als die Gießkanne.

Brot für die Welt
Postgiroamt Köln
500500500

TABELLE 2

–1 –8192	GRAM7 8 KBYTE	>FFFF >E000				
–8193 –16384	GRAM6 8KBYTE	>DFFF >C000				
–16385 –24576	GRAM 5 8 KBYTE	>BFFF >A000		GRAMBEREICH		
–24577 –32768	GRAM4 8 KBYTE	>9FFF >8000		RAMBEREICH		
32767 24576	GRAM 3 8 KBYTE	>7FFF >6000	32767 24576	RAM0 8 KBYTE	>7FFF >6000	

```
100 CALL CLEAR
110 CALL INIT
120 CALL LOAD("DSK1.BSCSUP")
130 CALL LOAD("DSK1.LOADER-E/A")
140 CALL CLEAR
150 CALL SCREEN(6)
160 FOR I=1 TO 16
170 CALL COLOR(I,16,6)
180 NEXT I
190 PRINT "   UNIVERSAL LOADER":"    MIT FILE EDITOR":" C 9/86 BY ZIEGLER":::
200 PRINT"   INPUT KOMMANDO ?"::
210 PRINT "   F FOR FILE LADEN(PROGRAM)"::
220 PRINT "   M FOR MODUL LADEN(D/V80)"::
230 PRINT "   E FOR EDIT FILE(D/V80)"::
240 PRINT "   Q FOR QUIT "::
250 A$=""
260 INPUT "   AKTION ? ":A$
270 IF A$="Q" THEN 900
280 IF A$="E" THEN 680
290 IF A$="M" THEN 470
300 IF A$="F" THEN 320
310 GOTO 260
320 CALL CLEAR
330 PRINT "   EINZEL-FILE "::"   LADEPROGRAM"::
340 PRINT
350 B$=""
360 INPUT "   FILENAME? DSK":B$
370 B$="DSK"&B$
380 CALL LINK("LOAD",B$)
390 PRINT
400 PRINT "   ** DONE **"::
410 PRINT "   F FOR FILE LADEN"::"   H FOR HAUPTMENUE"::" Q FOR QUIT"::
420 INPUT "   AKTION ? ":C$
430 IF C$="Q" THEN 900
440 IF C$="H" THEN 140
450 IF C$="F" THEN 320
460 GOTO 420
470 CALL CLEAR
480 PRINT "   LADEPROGRAM"::"   FUER MODULE":::
490 D$=""
500 INPUT "   FILENAME? DSK":D$
510 D$="DSK"&D$
520 OPEN #1:D$,DISPLAY ,VARIABLE 80, INPUT
530 FOR I=1 TO 6
540 PRINT
550 INPUT #1:E$(I)
560 IF SEG$(E$(I),1,3)="DSK" THEN 570 ELSE 590
570 PRINT "    "&E$(I)
580 CALL LINK("LOAD",E$(I))
590 NEXT I
600 CLOSE #1
610 PRINT
620 PRINT "   ** DONE **"::
630 PRINT "   H FOR HAUPTMENUE"::"   Q FOR QUIT"::
640 INPUT "   AKTION ? ":F$
650 IF F$="Q" THEN 900
660 IF F$="H" THEN 140
670 GOTO 640
```

```
680 CALL CLEAR
690 PRINT "   FILE EDITOR"::
700 G$=""
710 INPUT "   FILENAME? DSK":G$
720 G$="DSK"&G$
730 OPEN #1:G$,DISPLAY ,VARIABLE 80,OUTPUT
740 FOR I=1 TO 6
750 PRINT
760 INPUT "   FILE "&STR$(I)&"   "&"DSK":H$(I)
770 IF H$(I)="" THEN 790
780 H$(I)="DSK"&H$(I)
790 PRINT #1:H$(I)
800 NEXT I
810 CLOSE #1
820 PRINT ::
830 PRINT "   ** DONE **"::
840 PRINT "   E FOR FILE EDITOR"::"   H FOR HAUPTMENUE"::"  Q FOR QUIT"::
850 INPUT "   AKTION ? ":J$
860 IF J$="Q" THEN 900
870 IF J$="H" THEN 140
880 IF J$="E" THEN 680
890 GOTO 850
900 END
```

PERSONELLE ÄNDERUNGEN

Aus verschiedenen Gründen, insbesondere aber wegen Überlastung der einzelnen Departements, mußten einige personelle Änderungen im TI 99er Workshop Rheinland vorgenommen werden.

Zuständig für die Public-Domain und Freeware Bibliothek ist ab sofort:
TI 99er Workshop Rheinland
Dept. PD-Software
c/o Lothar Gergs
Geibelstr. 75
D-4000 Düsseldorf 1
0211/66 3743

Um Eure Hardwareprobleme und -fragen kümmert sich ab sofort:
TI 99er Workshop Rheinland
Dept. Hardware
c/o Bernhard Betz
Siegburger Str. 111
D-5000 Köln 21
02 21/88 45 21

Weiter: Ansprechpartner für allgemeine Informationen und Softwarefragen:
TI 99er Workshop Rheinland
Dept. Allgemein & Software
c/o Mike Heuser
Karl-Marx-Allee 18
D-5000 Köln 71
02 21/70 39 79

Bitte nur zwischen 19.00 und 22.00 Uhr anrufen und den Briefen Rückporto beilegen. Ihr schont so unsere Nerven und unsere schmale Portokasse. Auch die „neuen" Leute werden sich bemühen, Euch bei der Lösung Eurer Probleme und der Beantwortung Eurer Fragen, so weit und so schnell es geht, zu helfen.

TI 99er Workshop Rheinland
Mike Heuser

WIR HABEN EINEN TICK!

Man sagt uns TI-Usern nach, daß wir einen leichten Knacks weg haben müssen, wenn wir uns noch so lange mit dem „Oldie" unter den Computern beschäftigen. Wir Schleswig-Holsteiner bekennen uns: Wir haben einen TICK! Allerdings wollen wir damit nicht die Vorwürfe unserer lieben Kollegen bekräftigen; es handelt sich hierbei nämlich um die Abkürzung eines neuen Computer-Clubs, der sich ausschließlich mit dem TI-99/4A beschäftigt. Der TI-Club Kiel besteht zur Zeit noch aus 5 Mitgliedern, die alle aus Kiel kommen. Wir wollen aber einen Club bilden, an dem möglichst jeder TI-User aus Schleswig-Holstein beteiligt ist.

Aufgrund der räumlichen Nähe der bisherigen Mitglieder ist noch keine Clubzeitung geplant. Bei genügend Resonanz auf diesen Bericht könnte es allerdings zur Erstellung eines (regelmäßigen?) Magazins kommen.

Was wir aber in erster Linie wollen, ist folgendes: Wir wollen, daß jeder die Möglichkeiten seines texanischen Instrumentes besser kennenlernt und ausnutzt. (Uns ist es tatsächlich schon vorgekommen, daß jemand den TI mit einem Programm aus einer Zeitschrift nur als besseren Taschenrechner benutzt!) Zu diesem Zweck wollen wir jedem nach besten Möglichkeiten in der Programmierung in den einzelnen Sprachen des TIs unterstützen.

Eins soll aber klar sein: Wir wollen einen Club haben, der sich nur auf den Raum Schleswig-Holstein beschränkt! Dies hat natürlich auch seinen Grund; ab Elbe südwärts gibt es bereits genug Clubs, an die sich die TI-User aus diesen Regionen anschließen können. Im Raum S-H ist uns allerdings noch kein Club bekannt. Die räumliche Beschränkung soll auch das persönliche Kennenlernen der einzelnen Mitglieder fördern. Ein Clubtreffen, über das man dann auch reden könnte, wird dadurch natürlich auch bedeutend vereinfacht.

Wer sich für die Aktivitäten unseres Clubs interessiert, der kann sich an folgende Adresse wenden:
Michael Rittweger
Nissenstraße 12
2300 Kiel 14
Tel.: 04 31/7255 66

Bei schriftlichen Anfragen bitte unbedingt Rückporto beilegen, da wir noch keine Portokasse angelegt haben.

```
100 CALL CLEAR
110 CALL INIT
120 CALL LOAD("DSK1.MOVE")
130 PRINT " LADEPROGRAM FUER GPL":::
140 INPUT " FILE-NAME? ":A$
150 CALL LOAD(A$)
160 CALL LINK("MOVE")
170 PRINT ::
180 PRINT " GPL-PROGRAM INS GRAM";"        GELADEN"::
190 END
```

```
100 CALL CLEAR
110 PRINT "   PROGRAM ZUM SPEICHERN";"     VON BASICPROGRAMMEN ":"  INS GR
AM"
120 PRINT ::
130 CALL INIT
140 CALL LOAD("DSK1.BSCSUP")
150 CALL LOAD("DSK1.BASIC")
160 PRINT "   AUFRUF UEBER CALL LINK"
170 PRINT
180 END
```

FRAGEN AN DIE TI-REVUE

SCHREIBFEHLER IM PROGRAMM

In dem in TI-REVUE 4/87 vorgestellten Programm „Höhere Rechengenauigkeit" hat sich ein Fehler eingeschlichen. Daß das Programm so, wie abgedruckt, nicht laufen kann, dürfte auf der Hand liegen. Wir holen es deshalb nach; hier ist die korrigierte Version:

```
10INPUT "ZAEHLER,
  NENNER,STELLEN":
  A,B,S
30C=A/B :: C=INT(C)
  :: PRINT "C";",";
40FOR N=1 TO S
50D=B*C :: A= (A-D)
  :: C=A/B :: C=INT(C)
60PRINT C; :: NEXT N
```

PC-TASTATUR AN DEN TI

In Ihrer TI-REVUE 1/87 berichteten Sie unter anderem über die PC-Tastatur des Herrn Behnisch zum Anschluß an den TI-99. Trotz vieler Versuche ist es mir bis heute noch nicht gelungen, eine solche Tastatur zu erwerben. Da Herr Behnisch auf alle Fragen mit mehr oder weniger guten Ausreden reagierte, habe ich den Eindruck, daß an dieser Tastatur nicht sehr viel Wahres dran ist.
Detlef Keutgen, Grevenbroich

Die PC-Tastatur des Herrn Behnisch ist sowohl fertig als auch lauffähig. Der Vertrieb derselben liegt allein in der Hand von Herrn Behnisch. Es liegt also in seinem eigenen Ermessen, ob er schon verkaufen will oder noch nicht.

SELBSTBAU-PLOTTER FEHLERHAFT?

In TI-REVUE 2/87ff stellten Sie den Plotter zum Selbstbauen vor. Kann es sein, daß sich dort ein Fehler eingeschlichen hat, da bei mir einfach nichts läuft?
Rodolfo Bordenbue, Florida-Argentinien

Der Plotter ist fehlerfrei. Kontrollieren Sie am besten noch einmal alles nach, da sich bei einer solchen Schaltung sehr schnell Lötfehler etc. einschleichen.

UNBEKANNTES IM DISKKONTROLLER

Ich erforsche zur Zeit das Innere meines TI-Diskkontrollers und bin auf einige eigentümliche Adressen gestoßen, die im ROM liegen müßten, auf die aber geschrieben oder von denen gelesen wird. Es sind die Adressen >5FF0–>5FFE. Können Sie mir sagen, welche Bedeutung den einzelnen Adressen zukommt?
Michael Zapf, Karben

Der Adressbereich >5FF0–>5FFE enthält die Status-Register des Controllerbausteins, z.B. des WD1771, oder eines anderen. Diese sind ja, je nach Hersteller, verschieden.

DSR-MODUL FÜR DEN TI

Das DSR-Modul besteht aus 7 DSR-Adressen, die frei auf die Adressen >1000–>1F00 codiert werden können. Im ersten Ausbau stehen folgende Adressbereiche zur Verfügung: >1200,>1500, >1600, >1800, >1900, >1A00, >1B00.
Die Bestückung erfolgt, unter anderem, mit einem EPROM, einem 27128 und zwei RAMs, einem 6264 und einem 62256. Anstelle des 6264 kann man auch ein EPROM, ein 2764, einsetzen. Diese RAMs sind batteriegepuffert, so daß alle Daten auch nach Abschalten der P-Box erhalten bleiben.
Zwei „Bonbons" sind auch noch, in Form von Schaltern, eingebaut. Mit dem einen kann man die Versorgungsspannung des RAMs abschalten, also quasi ein Ein-Ausschalter für die Erweiterung.
Das zweite ist ein Schreibschutz. Es ist somit möglich, das RAM über einen Schalter zu einem Pseudo-ROM zu machen, von dem man nur noch lesen kann.
Die Stromversorgung besteht aus einer Batterie, an deren Stelle aber ohne Schwierigkeiten ein Accu eingesetzt werden kann.
Anfragen können direkt an mich gestellt werden:
**Eifried Martin
Albert Schöpfstr. 44
A-4020Linz**

ANDERE BAUDRATEN FÜR DIE RS232

In Heft 4/86 der TI-REVUE erschien ein Artikel, in dem beschrieben wurde, wie man die RS232 Schnittstelle auch an andere Baudraten anpassen kann. Nun galt dieser Artikel jedoch nur für die Original TI-Schnittstelle. Ich gehöre jedoch zu den Usern, die einen Nachbau, die RS232 von Atronic, besitzen. Die folgenden Anmerkungen beziehen sich nun auf die Schnittstellenkarte von Atronic, das EPROM enthält jedoch auch hier die Notiz "CPS99", so daß diese Applikation auch für das CPS99 verwendbar sein dürfte.

Fassen wir die Funktion doch noch einmal kurz zusammen. Auf der Schnittstellenkarte befinden sich zwei TMS9902 (für jeden Kanal einer), spezielle I/O-Bausteine für den TMS9900. Diese ICs bieten eine serielle Übertragung nach dem RS232/V24 Protokoll mit programmierbaren Parametern. Diese Parameter (Baudrate, Parity, Stoppbits, Zeichenlänge) werden in speziellen Registern im Baustein gespeichert, die direkt über die CRU angesprochen werden können. Da für Senden bzw. Empfangen verschiedene Baudraten gewählt werden können, sind auch hierfür zwei Register vorhanden, LRDR (Load Receive Data Rate) zum Empfangen, und LXDR (Load External Data Rate) zum Senden. Die Baudrate kann nun, dies ist von der DSR-Software der Schnittstelle so eingerichtet, nur in vorgewählten Werten programmiert werden, die in einer Tabelle gespeichert sind. Danach wird ein DATA-Wert ermittelt, der den passenden Teilfaktor, entsprechend Baudrate und interner Frequenz, angibt. Die Baudrate berechnet sich wie folgt: Das erste Bit, von TI CLK4M genannt, gibt an, ob die Taktfrequenz erst durch 3 (CLK4M=0) oder durch 4 geteilt werden muß. Das nächste zu beachtende Bit, Bit 5 des DATA-Wertes, wird von TI mit RDV8 (bzw. XDV8 für die Sendebaudrate) bezeichnet. Ist dieses Bit gesetzt, wird noch einmal durch 8 geteilt. Nun wird noch durch den Wert der Bits 6 bis 15 (RDR9–RDR0, bzw. XDR9–XDR0) geteilt. Schließlich muß das Ergebnis noch durch zwei geteilt werden, und man erhält die Baudrate. (Dieser Vorgang wird vom Programm im Listing erledigt.)

NUR IN VORGEWÄHLTEN WERTEN PROGRAMMIERBAR

Will man neue Baudraten implementieren, so muß man die Tabelle erweitern. Da diese jedoch mitten in der DSR Software liegt, beginnt man diese am besten nach der Software, ich habe sie ab Adresse >4A00 gelegt. Diese Tabellen wurden zwar in Heft 4/86 bereits veröffentlicht, leider enthielten sie dort einige Fehler, so daß die Baudrate nicht ganz korrekt gewählt wurde. Ich gebe deshalb noch einmal die Tabelle an. Hat man die Tabelle eingegeben, so muß man lediglich noch 3 Zeiger „verbiegen":
In Adresse >4596 steht die Adresse der ersten Baudrate, also> 4A02. In >4080 befindet sich ein Zeiger auf die Tabelle mit den Teilfaktoren für 2.5 MHz Takt, hier muß >4A34 stehen, in >4084 steht der Zeiger auf die Tabelle mit den Teilfaktoren für 3 MHz, hier muß nun >4A1C stehen. Hier noch einmal die Änderungen:

>4596 von >4086
zu> 4A02
>4080 von> 4098
zu >4A34 und
>4084 von >40A6
zu>4A1C

Natürlich muß für diese Änderungen die Software in einem freien RAM-Bereich verschoben werden, wie bereits beschrieben wurde. Hat man alles geändert, so kann man die Software in ein EPROM brennen. Hier haben wir Glück, Atronic verwendet ein 2764 EPROM, so daß man das neue DSR-ROM einfach austauschen kann. (Das Originaleprom sollte man jedoch tunlichst aufbewahren.) Nun müßte man auch mit dieser Schnittstelle Baudraten von 45 bis 31250 Baud „fahren" können.

Zum Schluß noch eine Anmerkung für die Benutzer der Original TI-Schnittstelle: Auch diese sollten die geänderten Werte für die Baudraten (natürlich auf den in 4/86 angegebenen Adressen) übernehmen, da dann die Baudrate besser mit dem Sollwert übereinstimmt (z.B. 1200 Bd, 3 MHz: alter Wert >01A4= 1190.48 Bd; neuer Wert >01A1 = 1199.04 Bd), wie man mit dem Programm leicht überprüfen kann.

HOMECOMPUTER AKTIV

MAILBOX

Tel.: 089 / 18 39 51

Parameter 8 N 1

24 Stunden Online!

```
100 C=2^15 :: F=3E6 ! INTERNE TAKTFREQUENZ
110 INPUT "DATA-WERT:":A$ :: CALL DEZ(A,A$):: IF A>=C THEN F=F/4 :: A=A-C ELSE F
=F/3
120 IF (A AND 1024)THEN F=F/8
130 F=F/2/(A AND 1023)
140 PRINT "BAUDRATE=";F :: GOTO 100
150 ! CALL DEZ AUS TI-REVUE 2/86
160 SUB DEZ(A,A$):: A$=SEG$("000"&A$,LEN(A$),4):: A=0 :: B=4096 :: FOR I=1 TO 4
:: B$=SEG$(A$,I,1):: A=A+B*(ASC(B$)-48+(B$>"=")*7):: B=B/16 :: NEXT I :: SUBEND
```

>4A00	=	>0000								
>4A02	=	>002D	>4A1C	=	>87FF	>4A34	=	>8764	(45)
>4A04	=	>0032	>4A1E	=	>87A8	>4A36	=	>870D	(50)
>4A06	=	>004B	>4A20	=	>8671	>4A38	-	>8609	(75)
>4A08	=	>006E	>4A22	=	>85AA	>4A3A	=	>8563	(110)
>4A0A	=	>012C	>4A24	=	>849C	>4A3C	=	>8482	(300)
>4A0C	=	>0258	>4A26	=	>8271	>4A3E	=	>8209	(600)
>4A0E	=	>04B0	>4A28	=	>01A1	>4A40	=	>015B	(1200)
>4A10	=	>0960	>4A2A	=	>809C	>4A42	=	>8082	(2400)
>4A12	=	>12C0	>4A2C	=	>804E	>4A44	=	>8041	(4800)
>4A14	=	>2580	>4A2E	=	>8027	>4A46	=	>002B	(9600)
>4A16	=	>4B00	>4A30	=	>001A	>4A48	=	>8402	(19200)
>4A18	=	>7A12	>4A32	=	>800C	>4A4A	=	>800A	(31250)
>4A1A	=	>0000							

```
1ØØ C=2^15 :: F=3E6 !INTERNE
 TAKTFREQUENZ
11Ø INPUT "DATA-WERT: ":A$ :
: CALL DEZ(A,A$): IF A>=C T
HEN F=F/4 :: A=A-C ELSE F=F/
3
12Ø IF (A AND 1Ø24)THEN F=F/
8
13Ø F=F/2/(A AND 1Ø23)
14Ø PRINT "BAUDRATE=";F :: G
OTO 1ØØ
15Ø ! CALL DEZ AUS TI-REVUE
2/86
16Ø SUB DEZ(A,A$):: A$=SEG$(
"ØØØ"&A$,LEN(A$),4):: A=Ø ::
 B=4Ø96 :: FOR I=1 TO 4 :: B
$=SEG$(A$,I,1): A=A+B*(ASC(
B$)-48+(B$>"=")*7):: B=B/16
:: NEXT I
17Ø SUBEND
```

EDITOR/ASSEMBLER ROUTINEN DISASSEMBLIERT

Hier nun der zweite Teil der Editor/Assembler-Routinen. Es ist für viele wohl der interessanteste von allen, nämlich der Loader.
Das Geheimnis, was diesen Loader wesentlich schneller als den XBASIC Loader macht ist, daß er in TMS9900 Assembler geschrieben wurde. Wie man ihn für das XBASIC nutzbar machen kann, werden wir in einer der nächsten TI-REVUES zeigen.

VERSANDLISTE IN EXTENDED BASIC-MLIST/XB-

Das Programm ist eine Ergänzung zum gleichlautenden Programmpaket von Texas Instruments und kann nur in Verbindung mit den weiteren Programmsegmenten dieses Paketes verwendet werden.

Jeder, der das Originalprogramm schon einmal verwendet hat, wird sehr bald feststellen, daß das Hauptprogramm "MLIST" sehr langsam läuft und umständlich zu bedienen ist. Das hier vorgestellte X-Basic-Programm läuft ca. 5mal schneller, es ist darüber hinaus voll kompatibel zum Basis-Programm, d.h., Sie können Ihre vorhandenen Dateien damit bearbeiten! Weiterhin wird der Suchlauf beim Aufrufen von Namen sicherer. Sollte ein Name nicht korrekt eingegeben sein, wiederholt das Programm den Suchlauf, wobei es nur die 3 ersten Buchstaben des Familiennamens und den ersten Buchstaben des Vornamens beachtet. Auch die Änderung der Gruppenzugehörigkeit benötigt nun nur noch Sekunden. Da nun auch Umlaute eingegeben werden können, sortiert das TI-Programm MLIST/CND unter Umständen nicht mehr richtig. Hierfür erforderliche geringe Änderungen am genannten Programm können beim Autor erfragt werden.
Geschrieben habe ich dieses Programm, weil ich umfangreiche Dateien dafür habe und mir der Zeitverlust bei der Bearbeitung derselben zu groß wurde.

Werner Stresemann
Gatowerstr. 83c
1000 Berlin 20
Tel.: 361 42 64

HUNGER
Kennwort „Hungerhilfe Afrika"
Menschen in Not brauchen Hilfe:
zuverlässig, schnell, wirksam. Die beiden kirchlichen Hilfswerke nehmen ihren Auftrag ernst.
Deutscher Caritasverband
Postgiro Karlsruhe 202
Diakonisches Werk
Postgiro Stuttgart 502
und viele Banken u. Sparkassen

```
* LOADER

23BA 04E0 CLR  @ULTAB     Entry Adresse loeschen
23BC 2022
23BE 53E0 SZCB @>20FC,R15 Loesche EQ-BIT in Calling Status
23C0 20FC
23C2 C020 MOV  @>8356,R0 PAB Pointer Namenslaenge
23C4 8356
23C6 0420 BLWP @DSRLNK    File eroeffnen
23C8 2120
23CA 0008 DATA >0008
23CC 1332 JEQ  >2432      ERROR ?
23CE 0220 AI   R0,>FFF7   PAB BYTE 0
23D0 FFF7
23D2 0201 LI   R1,>0200
23D4 0200
23D6 0420 BLWP @VSBW      Lies OP
23D8 210C
23DA 0580 INC  R0         Flag in FLGPTR
23DC C800 MOV  R0,@>202E
23DE 202E
23E0 C1E0 MOV  @>2024,R7 FSTHI in 7 (momentane Adresse)
23E2 2024
23E4 C147 MOV  R7,R5      und 5 (Relative Origin)
23E6 04CC CLR  R12
23E8 0680 BL   @>25E0    "DSRLNK" und aufbereiten
23EA 25E0
23EC 0283 CI   R3,>0001   TAG=CHR(1) (compressed Code) ?
23EE 0001
23F0 1624 JNE  >243A Nein,weiter mit TAGTest
23F2 058C INC  R12    Ja, setze Flag fuer compressed
23F4 04C3 CLR  R3     TAG loeschen
23F6 1023 JMP  >243E weiter ohne TAGTest

23F8 0283 CI   R3,>0046 TAG = 'F'
23FA 0046
23FC 161E JNE  >243A   Nein,weiter mit TAGTest

* Zeilenende

23FE 04C2 CLR  R2

* weiter nach EXEC TAG

2400 06A0 BL   @>262E naechstes TAG
2402 262E
2404 0283 CI   R3,>003A ':' ?
2406 003A
2408 16F7 JNE  >23F8     Nein, Test auf 'F'

* EOF

240A C020 MOV  @>202E,R0    PAB
240C 202E
240E 0600 DEC  R0         OP-CODE
2410 0201 LI   R1,>0100   :=1 (CLOSE)
2412 0100
2414 0420 BLWP @VSBW      schreibe
2416 210C
```

```
2418 Ø6AØ BL   @>25EØ     close File
241A 25EØ
241C CØ2Ø MOV  @ULTAB,RØ Startadresse vorhanden ?
241E 2Ø22
242Ø 13Ø7 JEQ  >243Ø      Nein,Ruechsprung
2422 Ø6AØ BL   @>2646     Noch REF's offen ?
2424 2646
2426 1ØØ5 JMP  >2432      Ja,Fehlermeldung
2428 CB4E MOV  R14,@>ØØ16(R13) alte Returnadresse in R11 calling WS
242A ØØ16
242C C3AØ MOV  @ULTAB,R14 speichere Startadresse
242E 2Ø22
243Ø Ø38Ø RTWP Ruecksprung, bzw EXEC

* Fehlerbehandlung

2432 D74Ø MOVB RØ,*R13    ERROR CODE in RØ Calling WS
2434 F3EØ SOCB @>2ØFC,R15 setze EQ-BIT in calling Status
2436 2ØFC
2438 Ø38Ø RTWP

* TAGTest

243A Ø6AØ BL   @>25C2     teste TAG,und gib in r3 ASCII zurueck
243C 25C2

243E Ø4C4 CLR  R4
244Ø D123 MOVB @>2662(R3),R4 Sprungadresse aus Tabelle
2442 2662                     in R4
2444 Ø974 SRL  R4,7          und Aufbereiten
2446 C8Ø8 MOV  R8,@>2Ø2C    Pruefsumme
2448 2Ø2C
244A Ø6AØ BL   @>2594       hole Byte
244C 2594
244E Ø464 B    @>23F8(R4)   weiter ausfuehren
245Ø 23F8

* Program ID

2452 Ø58Ø INC  RØ          Programmlaenge
2454 Ø24Ø ANDI RØ,>FFFE
2456 FFFE
2458 C12Ø MOV  @>2Ø24,R4 FSTHI in R4
245A 2Ø24
245C A1ØØ A    RØ,R4    Laenge addieren
245E 18Ø8 JOC  >247Ø    Overflow? dann Test im LOWMEM
246Ø 88Ø4 C    R4,@>2Ø26 zu lang?
2462 2Ø26
2464 1BØ5 JH   >247Ø      Ja, Teste Lowmwm
2466 C16Ø MOV  @>2Ø24,R5 Basis:=FSTHI
2468 2Ø24
246A C8Ø4 MOV  R4,@>2Ø24 FSTHI:=FSTHI+Programmlaenge
246C 2Ø24
246E 1ØØA JMP  >2484
247Ø C12Ø MOV  @>2Ø28,R4 FSTLOW
2472 2Ø28
2474 A1ØØ A    RØ,R4      Laenge addieren
2476 88Ø4 C    R4,@>2Ø2A  zu lang ?
```

```
2478 202A
247A 140C JHE   >2494      Ja, Fehlermeldung
247C C160 MOV   @>2028,R5 Basis:=FSTLOW
247E 2028
2480 C804 MOV   R4,@>2028 FSTLOW:=FSTLOW+Programmlaenge
2482 2028
2484 C1C5 MOV   R5,R7  Momentanadresse:=Basis

2486 0209 LI    R9,>0008 8 Bytes(I/D Name)
2488 0008
248A 06A0 BL    @>262E   Lesen
248C 262E
248E 0609 DEC   R9
2490 16FC JNE   >248A
2492      JMP   >2400   danach weiter

2494 0200 LI    R0,>0800   MEMORY OVERFLOW ERROR
2496 0800
2498      JMP   >2432

* Startadresse

249A A005 A     R5,R0      wenn relativ,Basis addieren
249C C800 MOV   R0,@ULTAB Startadresse speichern
249E 2022
24A0 10AF JMP   >2400      weiter

* Pruefsumme testen

24A2 A800 A     R0,@>202C Test durchfuehren
24A4 202C
24A6 13AC JEQ   >2400      OK,weiter
24A8 0200 LI    R0,>0B00   CHECKSUM ERROR
24AA 0B00
24AC 10C2 JMP   >2432      Fehler melden

* Adresse

24AE A005 A     R5,R0 wenn relativ, addiere Basis
24B0 C1C0 MOV   R0,R7 Adresse schreiben
24B2 10A6 JMP   >2400 weiter

* DATA schreiben

24B4 A005 A     R5,R0 wenn relativ, Basis addieren
24B6 DDC0 MOVB  R0,*R7+ schreibe HIGH Byte
24B8 DDE0 MOVB  @>20DB,*R7+  schreibe LOW Byte
24BA 20DB
24BC 10A1 JMP   >2400 weiter

* REF

24BE A005 A     R5,R0  wenn relativ, addiere Basis
24C0 06A0 BL    @>2566 in REF/DEF Table schreiben
24C2 2566
24C4 C000 MOV   R0,R0  Adresse=0? ( nur geREFt, aber nicht verwendet)
24C6 1316 JEQ   >24F4  Ja, Eintrag rauswerfen und Exit
```

```
* Suche passendes DEF

24C8 0226 AI   R6,-8  Naechster Eintrag in REF
24CA FFF8
24CC 8106 C    R6,R4  Alle durch ?
24CE 1B02 JH   >24D4
24D0 0514 NEG  *R4    Ja, markiere Eintrag als REF
24D2 1096 JMP  >2400  weiter im Loader
24D4 8594 C    *R4,*R6  Sind die Eintraege gleich ?
24D6 16F8 JNE  >24C8    Nein
24D8 89A4 C    @>0002(R4),@>0002(R6)
24DA 0002
24DC 0002
24DE 16F4 JNE  >24C8    Nein
24E0 89A4 C    @>0004(R4),@>0004(R6)
24E2 0004
24E4 0004
24E6 16F0 JNE  >24C8    Nein
24E8 C0E6 MOV  @>0006(R6),R3 Def gefunden,
24EA 0006                    alle REF's abaendern
24EC C250 MOV  *R0,R9
24EE C403 MOV  R3,*R0
24F0 C009 MOV  R9,R0
24F2 16FC JNE  >24EC      alle  durch ?
24F4 0224 AI   R4,>0008   LSTLOW:=LSTLOW+4
24F6 0008                 (= REF Eintrag loeschen)
24F8 C804 MOV  R4,@>202A  speichere LSTLOW
24FA 202A
24FC 10EA JMP  >24D2      (= JMP >2400) weiter im Loader

* DEF

24FE A005 A    R5,R0      wenn relativ, addiere Basis
2500 06A0 BL   @>2566     In REF/DEF Table schreiben
2502 2566

* gehe alle Eintraege durch

2504 0226 AI   R6,-8      Naechster Eintrag
2506 FFF8
2508 8106 C    R6,R4      alle durch ?
250A 13E3 JEQ  >24D2      (= JMP >2400) weiter im Loader
250C C296 MOV  *R6,R10    Vergleiche Namen
250E 1501 JGT  >2512
2510 050A NEG  R10        REF-Eintrag , da erstes Byte negiert
2512 8294 C    *R4,R10    Namen gleich ?
2514 16F7 JNE  >2504      Nein
2516 89A4 C    @>0002(R4),@>0002(R6)
2518 0002
251A 0002
251C 16F3 JNE  >2504      Nein
251E 89A4 C    @>0004(R4),@>0004(R6)
2520 0004
2522 0004
2524 16EF JNE  >2504      Nein
2526 C296 MOV  *R6,R10    Namen sind gleich. War Eintrag REF ?
2528 1516 JGT  >2556      Nein, DUPLICATE DEFINITION ERROR
252A C0E6 MOV  @>0006(R6),R3   REF Adresse
```

```
252C Ø006
252E C253 MOV  *R3,R9
253Ø C4CØ MOV  RØ,*R3       durch DEF-Adresse ersetzen
2532 CØC9 MOV  R9,R3        alle durch ?
2534 16FC JNE  >252E        Nein, weitermachen
2536 C246 MOV  R6,R9        REF-Eintrag loeschen
2538 6244 S    R4,R9
253A C286 MOV  R6,R1Ø       und Table 'Crunchen'
253C Ø22A AI   R1Ø,>ØØØ8
253E ØØØ8
254Ø CØC6 MOV  R6,R3
2542 Ø643 DECT R3
2544 Ø64A DECT R1Ø
2546 C693 MOV  *R3,*R1Ø
2548 Ø649 DECT R9
254A 16FB JNE  >2542
254C Ø224 AI   R4,>ØØØ8     LSTLOW:=LSTLOW+4
254E ØØØ8
255Ø C8Ø4 MOV  R4,@>2Ø2A    speichern
2552 2Ø2A
2554 1ØD9 JMP  >25Ø8        Naechsten Eintrag testen

2556 CB44 MOV  R4,@>ØØØ2(R13) in R1 calling WS
2558 ØØØ2
255A Ø2ØØ LI   RØ,>ØCØØ     DUPLICATE DEFINITION ERROR
255C ØCØØ
255E Ø46Ø B    @>2432       Fehler melden
256Ø 2432
2562 Ø46Ø B    @>2494       MEMORY OVERFLOW ERROR melden
2564 2494

* Eintrag in REF/DEF Table schreiben

2566 C28B MOV  R11,R1Ø      Save RT Adresse
2568 Ø2Ø9 LI   R9,>ØØØ6     6 Byte Namenslaenge
256A ØØØ6
256C C1AØ MOV  @>2Ø2A,R6    Eintrag anhaengen
256E 2Ø2A
257Ø Ø226 AI   R6,-8
2572 FFF8
2574 C1Ø6 MOV  R6,R4        neues LSTLOW zwischenspeichern
2576 88Ø6 C    R6,@>2Ø28    LSTLOW < FSTLOW
2578 2Ø28
257A 1AF3 JL   >2562        Ja, MEMORY OVERFLOW
257C C8Ø6 MOV  R6,@>2Ø2A    sonst LSTLOW setzen
257E 2Ø2A
258Ø Ø6AØ BL   @>262E       Namen lesen
2582 262E
2584 DDAØ MOVB @>2ØE1,*R6+  und schreiben
2586 2ØE1
2588 Ø6Ø9 DEC  R9
258A 16FA JNE  >258Ø
258C C58Ø MOV  RØ,*R6       Adresse in Table schreiben
258E Ø2Ø6 LI   R6,>4ØØØ     Absolutes Ende LOWMEM
259Ø 4ØØØ
2592 Ø45A B    *R1Ø     quasi RT

* 2 Byte aus Tagged Code holen und in RØ
```

```
2594 C28B MOV  R11,R10     Save RT Adresse
2596 04C0 CLR  R0
2598 C30C MOV  R12,R12     compressed Code ?
259A 1308 JEQ  >25AC       Ja, nur 2 Bytes jesen
259C 06A0 BL   @>262E      Byte lesen
259E 262E
25A0 D020 MOVB @>20E1,R0
25A2 20E1
25A4 06A0 BL   @>262E      Byte lesen
25A6 262E
25A8 A003 A    R3,R0       Addieren
25AA 045A B    *R10        zurueck

25AC 0209 LI   R9,>0004    4 Bytes lesen
25AE 0004
25B0 06A0 BL   @>262E      byte lesen
25B2 262E
25B4 06A0 BL   @>25C2      Byte testen und umwandeln
25B6 25C2
25B8 0A40 SLA  R0,4
25BA A003 A    R3,R0
25BC 0609 DEC  R9
25BE 16F8 JNE  >25B0       alle durch ?
25C0 045A B    *R10        Ja, RT

* Test auf korrektes TAG und umwandeln TAG -> Wert

25C2 0223 AI   R3,>FFD0    ASCII 0 abziehen
25C4 FFD0
25C6 0283 CI   R3,>000A WENN Ziffer, dann OK
25C8 000A
25CA 1A05 JL   >25D6
25CC 0223 AI   R3,>FFF9    sonst Buchstabe: nochmal 7 abziehen
25CE FFF9
                           TAG >'I' ? Test
25D0 0283 CI   R3,>0019    Fehler im ROM !! muss Dez 19 (0-I) sein !!!
25D2 0019                  Hier wurde von TI nicht beruecksichtigt, dass
                           bereits 7 im Falle eines Buchstabe abgezogen
                           wurden.
25D4 1B01 JH   >25D8       Falsches TAG
25D6 045B RT           Sonst RT

25D8 0200 LI   R0,>0A00 BAD TAG ERROR
25DA 0A00
25DC 0460 B    @>2432    Fehler melden
25DE 2432

* "DSRLNK" und Aufbereitung

25E0 02E0 LWPI GPLWS
25E2 83E0
25E4 0200 LI   R0,>2032    SAVCRU
25E6 2032
25E8 C330 MOV  *R0+,R12 CRU Adr. IN 12
25EA C270 MOV  *R0+,R9   ENTRY IN 9
25EC C830 MOV  *R0+,@>8354 NAMENLAENGE
25EE 8354
25F0 C830 MOV  *R0+,@>8356 PAB Pointer setzen
```

```
25F2 8356
25F4 C050 MOV  *R0,R1 SAVVER
25F6 1D00 SBO  0 ROM anschalten
25F8 9820 CB   @>4000,@>20FF DSR noch da ?
25FA 4000
25FC 20FF
25FE 161D JNE  >263A  Nein,I/O ERROR
2600 0699 BL   *R9    EXEC DSR
2602 101B JMP  >263A  Fehler bei der Ausfuehrung => I/O ERROR
2604 1E00 SBZ  0      ROM AUS
2606 02E0 LWPI WS#4   altes WS
2608 20DA
260A C020 MOV  @>202E,R0 FLGPTR
260C 202E
260E 0201 LI   R1,>20DB BUFFER:=WS+1
2610 20DB
2612 0202 LI   R2,>0004
2614 0004
2616 0420 BLWP @VMBR     lese PAB Byte 1-4
2618 2118
261A 7000 SB   R0,R0     loesche oberes Byte
261C 0950 SRL  R0,5      Fehler passiert ?
261E 1610 JNE  >2640     Ja, Error
2620 0982 SRL  R2,8      Datensatzlaenge in R2
2622 C001 MOV  R1,R0     DATA Buffer im VDP
2624 0201 LI   R1,>203C  neuer Buffer
2626 203C
2628 0420 BLWP @VMBR    Lese Datensatz
262A 2118
262C 04C8 CLR  R8    CHKSUM:=0

* Byte lesen

262E 0602 DEC  R2
2630 11D7 JLT  >25E0  wenn alle gelesen, dann neuen Record lesen
2632 D0F1 MOVB *R1+,R3 Byte lesen
2634 0983 SRL  R3,8    in R3
2636 A203 A    R3,R8   und zur CHKSUM in R8 addieren
2638 045B RT          und RT

* Fehler melden

263A 02E0 LWPI WS#4   bei Bedarf WS laden
263C 20DA
263E 04C0 CLR  R0     I/O ERROR 00
2640 06C0 SWPB R0     Error Code in HIGH-Byte
2642 0460 B    @>2432 Fehler melden
2644 2432

* Test auf offene REF's

2646 0201 LI   R1,>3F40 Die vordefinierten DEF's liegen ab 3F40 im RAM
2648 3F40
264A 0221 AI   R1,-8    naechster Eintrag
264C FFF8
264E C011 MOV  *R1,R0 WENN REF dann
2650 1105 JLT  >265C  Fehler melden
2652 8060 C    @>202A,R1 Ende REF/DEF Table ?
```

```
2654 2Ø2A
2656 16F9 JNE   >264A Nein naechster Eintrag
2658 Ø5CB INCT R11    1 Befehl Skip, da kein Fehler
265A Ø45B RT        und RT

265C Ø2ØØ LI    RØ,>ØDØØ  UNRESOLVED REFERENCE ERROR
265E ØDØØ
266Ø Ø45B RT     Zurueck mit FEhlermelden

* Tabelle der TAG-Routinen
* Die mit * gekennzeichnetet verursachen bei diesem Loader
* eine Fehlermeldung

2662 2D52 BYTE >2D Ø IDENTIFY
               >52 1 ABSOLUTE ENTRY
2664 5163      >51 2 RELATIVE    "
               >63 3 RELATIVE REF
2666 6483      >64 4 ABSOLUTE  "
               >83 5 ABSOLUTE DEF
2668 8455      >84 6 RELATIVE  "
               >55 7          CHECKSUM
266A Ø45C      >Ø4 8 IGNORE      "
               >5C 9 ABSOLUTE ADRESSE
266C 5B5F      >5B A RELATIVE   "
               >5F B ABSOLUTE DATEN
266E 5EFØ      >5E C RELATIVE    "
               >FØ D *
267Ø FØØ3      >FØ E *
               >Ø3 F ENDE ZEILE
2672 FØFØ      >FØ G *
               >FØ H *
2674 47        >47 Ignore
       ØØ EVEN
2676 FFØØ DATA >FFØØ        1.Freie Adresse in LM
≡-
)UUUUUUUUUU
```

IMPRESSUM

HCA (MSX&TI-REVUE),die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Geschäftsführer: Werner E. Seibt
Redaktion: Alfons Bauer, Thomas Dehn, Jürgen Puschmann, Uwe Schröder
Verantwortlich für den Inhalt:
MSX-Teil: Werner E. Seibt
TI-Teil: Oliver Huber (Technik) Hartmut König (Listings)
Tel.: 0 89/ 129 80 11
Telex: 5214428 cav-d
Alle: Postfach 1161
8044 Lohhof.
Anfragen bitte nur schriftlich
Druck: Dierichs, Kassel
Es gilt die Honorarliste des Verlages.

Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Tiexten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.

Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

© by CA Verlag, Heßstraße 90, 8000 München 40 SPS und Autoren

Erstverkaufstag: 31.7.1987

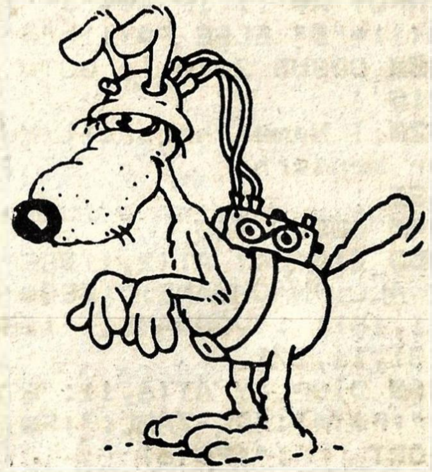
```
100 !************************
110 !*                      *
120 !*     VERSANDLISTE     *
130 !*       MLIST/XB       *
140 !*                      *
150 !*  Copyright 1987 by   *
160 !*  Werner Stresemann   *
170 !*                      *
180 !* X-BASIC-Programm zum*
190 !*  Programmpaket von   *
200 !*  Texas-Instruments   *
210 !*                      *
220 !* Benoetigte Geraete:  *
230 !*    TI99/4A-Konsole   *
240 !*      EXT.-BASIC      *
250 !* 1 Diskettenlaufwerk  *
260 !*       Optionen:      *
270 !*        Drucker       *
280 !*  Speicherbelegung:   *
290 !*      8997 BYTES      *
300 !************************
310 !
320 DIM F$(13)
330 !
340 ! Umlaute
350 !
360 CALL CHAR(126,"0038444458444458"):: KR$="00440038447C4444"
370 CALL CHAR(91,KR$):: CALL CHAR(123,KR$)
380 KR$="0044003844444438" :: CALL CHAR(92,KR$):: CALL CHAR(124,KR$)
390 KR$="0044004444444438" :: CALL CHAR(93,KR$):: CALL CHAR(125,KR$)
400 !
410 ! Progamm-Start
420 !
430 DISPLAY AT(6,2)BEEP ERASE ALL:"V E R S A N D L I S T E "
440 DISPLAY AT(10,1):"DATENDISKETTE IN ANTRIEB 1"
450 GOSUB 460 :: GOTO 480
460 DISPLAY AT(24,5):"WEITER MIT LEERTASTE" :: CALL KEY(0,K,S):: IF S=0 OR K<>32 THEN 460
470 RETURN
480 OPEN #1:"DSK1.",RELATIVE,INTERNAL,INPUT
490 INPUT #1:A$,A,A,A,B$,B,B
500 IF B$="M_LIST/DAT" THEN CLOSE #1 :: GOTO 540
510 IF A$<>"" THEN 490
520 DISPLAY AT(10,1)BEEP ERASE ALL:"DATEI NICHT AUF DISKETTE !" :: CLOSE #1 :: GOSUB 460 :: GOTO 440
530 !
540 OPEN #1:"DSK1.M_LIST/DAT",RELATIVE,INTERNAL,FIXED 255
550 INPUT #1,REC 0:LSP,LP,BUS$,GEN$,DEV$,K$(1),K$(2),K$(3),K$(4),K$(5),K$(6),K$(7),K$(8),K$(9),K$(10),NK,NL,PN$
560 BLANK$="                            "
570 !
580 !
590 DISPLAY AT(6,8)BEEP ERASE ALL:A$: : :"DATENS[TZE EINSORTIERT:";LSP: :"      NICHT EINSORTIERT:";LP-LSP :: GOSUB 460
600 !
610 ! Wahl-Liste 1
620 !
630 DISPLAY AT(6,8)BEEP ERASE ALL:"VERSANDLISTE": : :"DATENS[TZE LESEN   => 1": :"DATENS[TZE [NDERN => 2": :"DATEI GRUPPIEREN  => 3"
640 DISPLAY AT(16,1):"ENDE              => 4": : :"BITTE W[HLEN SIE:"
650 ACCEPT AT(19,19):X :: IF X<1 OR X>4 THEN 630 ELSE ON X GOTO 700,1080,1630,660
660 CLOSE #1 :: END
670 !
680 ! Wahl-liste 2
690 !
700 DISPLAY AT(6,6)BEEP ERASE ALL:"DATENS[TZE LESEN": : :"EINZELPERSON    => 1": :"GRUPPEN        => 2": :"GANZE DATEI    => 3": :"R]CKSPRUNG     => 4"
710 DISPLAY AT(18,1):"BITTE W[HLEN SIE" :: ACCEPT AT(18,19):X :: IF X<1 OR X>4 THEN 700 ELSE ON X GOTO 750,850,1030,630
720 !
730 ! Einzel-Person lesen
740 !
750 GOSUB 760 :: GOSUB 910 :: GOSUB 460 :: GOTO 630
760 DISPLAY AT(6,1)BEEP ERASE ALL:"BITTE DEN NAMEN EINGEBEN !": : :"NACHNAME": :"VORNAME"
```

```
77Ø ACCEPT AT(9,1Ø)SIZE(15):
NA$ :: ACCEPT AT(11,1Ø)SIZE(
12):VOR$ :: GOSUB 78Ø :: FLA
G=Ø :: GOTO 79Ø
78Ø N$=SEG$(NA$&BLANK$,1,15)
&SEG$(VOR$&BLANK$,1,12):: RE
TURN
79Ø GOSUB 18ØØ
8ØØ IF A<>Ø THEN RETURN ELSE
 IF FLAG=Ø THEN FLAG=1 :: GO
TO 79Ø
81Ø DISPLAY AT(1Ø,1)BEEP ERA
SE ALL:"PERSON NICHT IN DER
DATEI!" :: GOSUB 46Ø :: GOTO
 63Ø
82Ø !
83Ø ! Personen-Gruppen lesen
84Ø !
85Ø DISPLAY AT(1,3)BEEP ERAS
E ALL:"GRUPPEN-BEZEICHNUNGEN
:": : :"1- ";K$(1): :"2- ";K
$(2): :"3- ";K$(3): :"4- ";K
$(4): :"5- ";K$(5)
86Ø X,Y=Ø
87Ø DISPLAY AT(14,1):"6- ";K
$(6): :"7- ";K$(7): :"8- ";K
$(8): :"9- ";K$(9): :"1Ø-";K
$(1Ø): :"WELCHE GRUPPE ?:"
88Ø ACCEPT AT(24,18):N :: FO
R Z=1 TO LP :: GOSUB 199Ø
89Ø IF SEG$(F$(2),N,1)="J" T
HEN GOSUB 97Ø
9ØØ NEXT Z :: DISPLAY AT(1Ø,
5)BEEP ERASE ALL:"DATEN VOLL
ST[NDIG" :: GOSUB 46Ø :: GOT
O 63Ø
91Ø PRINT :F$(13):F$(3);:F$(
4);:F$(5);F$(6);:F$(7):F$(8)
: :F$(9):F$(1Ø):F$(11):F$(12
):: PRINT
92Ø FLAG=Ø
93Ø FOR A=1 TO 1Ø :: IF SEG$
(F$(2),A,1)="N" THEN 94Ø ELS
E PRINT K$(A):: FLAG=1
94Ø NEXT A :: PRINT
95Ø IF F$(1)="D" THEN PRINT
"L\SCHEN" ELSE IF FLAG=Ø THE
N PRINT "INAKTIV" :: F$(1)="
I"
96Ø PRINT :: RETURN
97Ø X=2 :: DISPLAY AT(X,1)ER
ASE ALL:F$(3): FOR Y=1 TO 1
Ø :: IF SEG$(F$(2),Y,1)="N"
THEN 99Ø
98Ø X=X+2 :: DISPLAY AT(X,1)
:K$(Y)
99Ø NEXT Y :: GOSUB 46Ø :: R
ETURN
```

```
1ØØØ !
1Ø1Ø ! Alle Saetze lesen
1Ø2Ø !
1Ø3Ø CALL CLEAR :: FOR Z=1 T
O LP :: GOSUB 197Ø :: PRINT
Z;" ";F$(3)
1Ø4Ø PRINT :: GOSUB 46Ø :: N
EXT Z :: PRINT :: PRINT "
   DATEN VOLLST[NDIG" :: GOT
O 63Ø
1Ø5Ø !
1Ø6Ø ! Daten {ndern
1Ø7Ø !
1Ø8Ø DISPLAY AT(2,1)BEEP ERA
SE ALL:"PERSON HINZUF]GEN
     => 1": :"PERSON L\SCHEN
          => 2": :"NAME/ANSCH
RIFT/"
1Ø9Ø DISPLAY AT(7,1):"KOMMEN
TAR        [NDERN => 3": :"G
RUPPENZUORDNUNG [NDERN => 4"
: :"R]CKSPRUNG
=> 5"
11ØØ DISPLAY AT(15,1):"BITTE
 W[HLEN SIE :"
111Ø ACCEPT AT(15,2Ø)VALIDAT
E(DIGIT):X :: IF X<1 OR X>5
THEN 1Ø8Ø ELSE ON X GOTO 112
Ø,138Ø,144Ø,153Ø,63Ø
112Ø !
113Ø ! Person hinzufuegen
114Ø !
115Ø LP=LP+1 :: Z=LP :: GOSU
B 116Ø :: GOSUB 118Ø :: GOTO
 125Ø
116Ø DISPLAY AT(2,1)BEEP ERA
SE ALL:"NACHNAME:": :"VORNAM
E :": :"STRASSE:": :"PL-ZAHL
:": :"ORT     :": :"LAND   :"
: :"TELEFON:"
117Ø DISPLAY AT(16,1):"KOMT.
1:": :"KOMT.2:": :"KOMT.3:":
 :"KOMT.4:": :"ANREDE :" ::
RETURN
118Ø ACCEPT AT(2,1Ø)SIZE(-15
):NA$ :: ACCEPT AT(4,1Ø)SIZE
(-12):VOR$ :: ACCEPT AT(7,1)
SIZE(-27):F$(4):: ACCEPT AT(
8,9)SIZE(-7):F$(5)
119Ø ACCEPT AT(1Ø,9)SIZE(-2Ø
):F$(6):: ACCEPT AT(12,9)SIZ
E(-15):F$(7):: ACCEPT AT(14,
9)SIZE(-2Ø):F$(8)
12ØØ ACCEPT AT(16,9)SIZE(-18
):F$(9):: ACCEPT AT(18,9)SIZ
E(-18):F$(1Ø):: ACCEPT AT(2Ø
,9)SIZE(-18):F$(11)
121Ø ACCEPT AT(22,9)SIZE(-18
```

```
):F$(12) :: ACCEPT AT(24,9)SI
ZE(-5):F$(13)
122Ø FOR X=4 TO 13 :: IF X=4
 OR X=9 OR X=1Ø OR X=11 OR X
=12 THEN Y=27 ELSE IF X=5 TH
EN Y=7 ELSE IF X=7 THEN Y=15
123Ø IF X=6 OR X=8 THEN Y=2Ø
 ELSE IF X=13 THEN Y=5
124Ø F$(X)=SEG$(F$(X)&BLANK$
,1,Y):: NEXT X :: RETURN
125Ø DISPLAY AT(1,1)BEEP ERA
SE ALL:"GRUPPENZUORDNUNG J /
 N": :K$(1): :K$(2): :K$(3):
 :K$(4): :K$(5): :K$(6): :K$
(7): :K$(8): :K$(9): :K$(1Ø)
126Ø X=3 :: Y=1
127Ø IF X>=22 THEN 129Ø ELSE
 ACCEPT AT(X,2Ø)VALIDATE("JN
jn"):G$(Y):: IF G$(Y)="j" TH
EN G$(Y)="J" ELSE IF G$(Y)="
n" THEN G$(Y)="N"
128Ø X=X+2 :: Y=Y+1 :: GOTO
127Ø
129Ø FOR X=1 TO 1Ø :: IF G$(
X)="" THEN G$(X)="N"
13ØØ F$(2)=F$(2)&G$(X):: NEX
T X :: GOSUB 78Ø :: F$(3)=N$
 :: IF F$(2)="NNNNNNNNNN" TH
EN F$(1)="I" ELSE F$(1)="A"
131Ø GOSUB 2Ø1Ø
132Ø DISPLAY AT(1Ø,1)BEEP ER
ASE ALL:"MEHR HINZUF]GEN => 
1": :"R]CKSPRUNG        => 2":
 :"BITTE W[HLEN SIE:"
133Ø ACCEPT AT(14,18):X :: I
F X<>1 THEN GOSUB 2Ø5Ø :: GO
TO 59Ø
134Ø GOTO 115Ø
135Ø !
136Ø ! Person loeschen
137Ø !
138Ø GOSUB 76Ø :: GOSUB 91Ø
139Ø PRINT "L\SCHEN? J/N" ::
 INPUT A$ :: IF A$="J" THEN
F$(1)="D" ELSE F$(1)="A"
14ØØ GOSUB 2Ø1Ø :: GOTO 63Ø
141Ø !
142Ø ! Name/Anschrift/Kommen
tar aendern
143Ø !
144Ø GOSUB 76Ø
145Ø DISPLAY AT(2,1)BEEP ERA
SE ALL:"NACHNAME:";SEG$(F$(3
),1,15): :"VORNAME: ";SEG$(F
$(3),16,27)
146Ø DISPLAY AT(6,1):"STRASS
E:";F$(4):"PL-ZAHL:";F$(5):
:"ORT     :";F$(6)
147Ø DISPLAY AT(12,1):"LAND
 :";F$(7): :"TELEFON:";F$(8
)
148Ø DISPLAY AT(16,1):"SEGEL
-#:";SEG$(F$(9),1,1Ø): :"CLU
B    :";SEG$(F$(1Ø),1,1Ø): :"
FLOTTE :";SEG$(F$(11),1,1Ø)
149Ø GOSUB 118Ø :: GOSUB 78Ø
 :: F$(3)=N$ :: GOSUB 2Ø1Ø :
: GOTO 1Ø8Ø
15ØØ !
151Ø ! Gruppenzuordnung aend
ern
152Ø !
153Ø GOSUB 76Ø
154Ø DISPLAY AT(1,1)BEEP ERA
SE ALL:F$(3):: FOR X=1 TO 1Ø
 :: DISPLAY AT(2*X+1,1):K$(X
):: DISPLAY AT(2*X+1,2Ø):SEG
$(F$(2),X,1)
155Ø NEXT X :: F$(2)=""
156Ø FOR X=1 TO 1Ø :: ACCEPT
 AT(2*X+1,2Ø)SIZE(-2)VALIDAT
E("JNjn"):G$(X):: IF G$(X)="
j" THEN G$(X)="J" ELSE IF G$
(X)="n" THEN G$(X)="N"
157Ø F$(2)=F$(2)&G$(X)
158Ø NEXT X :: IF F$(2)="NNN
NNNNNNN" THEN F$(1)="I" ELSE
 F$(1)="A"
159Ø GOSUB 2Ø1Ø :: GOTO 1Ø8Ø
16ØØ !
161Ø ! Datei gruppieren
162Ø !
163Ø DISPLAY AT(4,7)BEEP ERA
SE ALL:"DATEI GRUPPIEREN": :
 : :"GRUPPEN BILDEN    => 1":
 :"GRUPPEN L\SCHEN   => 2": :
"R]CKSPRUNG          => 3"
1635 DISPLAY AT(18,5):"BITTE
 W[HLEN SIE :"
164Ø ACCEPT AT(18,25)VALIDAT
E(DIGIT):X :: IF X<1 OR X>3
THEN 163Ø ELSE ON X GOTO 165
Ø,179Ø,63Ø
165Ø GOSUB 166Ø :: GOSUB 168
Ø :: GOTO 17ØØ
166Ø DISPLAY AT(1,1)BEEP ERA
SE ALL:"1- ";K$(1): :"2- ";K
$(2): :"3- ";K$(3): :"4- ";K
$(4): :"5- ";K$(5)
167Ø DISPLAY AT(11,1):"6- ";
K$(6): :"7- ";K$(7): :"8- ";
K$(8): :"9- ";K$(9): :"1Ø-";
K$(1Ø):: RETURN
168Ø DISPLAY AT(22,1):"BITTE
 GRUPPEN-NO. EINGEBEN": :"R]
CKSPRUNG   => 11"
```

```
169Ø ACCEPT AT(24,24)VALIDAT
E(DIGIT):X :: IF X<1 OR X>11
 THEN 163Ø ELSE IF X=11 THEN
 63Ø ELSE RETURN
17ØØ IF K$(X)<>" " THEN 174Ø
171Ø DISPLAY AT(6,1)BEEP ERA
SE ALL:"NEUER GRUPPEN-NAME:"
 :: ACCEPT AT(8,1):K$(X):: N
K=NK+1 :: GOSUB 2Ø5Ø
1712 DISPLAY AT(12,1):"GANZE
 DATEI ZUORDNEN? J/N" :: ACC
EPT AT(12,27)VALIDATE("JN"):
IN$ :: IF IN$="N" THEN 172Ø
1714 DISPLAY AT(6,8)BEEP ERA
SE ALL:"BITTE WARTEN!": : :"
  DATEI WIRD KORRIGIERT"
1716 FOR Z=1 TO LP :: GOSUB
199Ø :: F$(2)=SEG$(F$(2),1,X
-1)&IN$&SEG$(F$(2),X+1,1Ø)::
 GOSUB 2Ø1Ø :: NEXT Z
172Ø DISPLAY AT(12,1):"MEHR
HINZUF]GEN? J/N" :: ACCEPT A
T(12,22)VALIDATE("JnjN"):X$
:: IF X$<>"J" AND X$<>"j" TH
EN 63Ø ELSE 165Ø
173Ø !
174Ø DISPLAY AT(6,1)BEEP ERA
SE ALL:"GRUPPE BELEGT! [NDER
N? J/N" :: ACCEPT AT(6,27)VA
LIDATE("JjnN"):X$ :: IF X$<>
"J" AND X$<>"j" THEN 63Ø
175Ø GOSUB 176Ø :: GOTO 178Ø
176Ø DISPLAY AT(6,8)BEEP ERA
SE ALL:"BITTE WARTEN!": : :"
  DATEI WIRD KORRIGIERT"
177Ø K$(X)=" " :: NK=NK-1 ::
 FOR Z=1 TO LP :: GOSUB 199Ø
 :: F$(2)=SEG$(F$(2),1,X-1)&
"N"&SEG$(F$(2),X+1,1Ø):: GOS
UB 2Ø1Ø :: NEXT Z :: RETURN
178Ø GOSUB 2Ø5Ø :: GOTO 171Ø
179Ø GOSUB 166Ø :: DISPLAY A
T(21,1)"WELCHE GRUPPE M\CHT
EN": :"SIE L\SCHEN?" :: GOSU
B 169Ø :: GOSUB 176Ø :: GOSU
B 2Ø5Ø :: GOTO 63Ø
18ØØ !
181Ø ! Suchen
182Ø !
183Ø A=Ø :: I=1 :: J=LSP
184Ø Z=INT((I+J)/2):: GOSUB
197Ø :: IF FLAG=1 THEN GOSUB
 193Ø
185Ø IF F$(3)=N$ THEN 19ØØ E
LSE IF F$(3)<N$ THEN 186Ø EL
SE J=Z-1 :: GOTO 187Ø
186Ø I=Z+1
187Ø IF (Z<1)+(I>J)=Ø THEN 1
84Ø ELSE IF LSP=LP THEN 192Ø
188Ø FOR Z=LSP+1 TO LP :: GO
SUB 197Ø :: IF F$(3)=N$ THEN
 19ØØ
189Ø NEXT Z :: Z=Ø
19ØØ A=Z :: IF Z=Ø THEN 192Ø
191Ø GOSUB 199Ø
192Ø RETURN
193Ø F$(3)=SEG$(F$(3),1,4)&S
EG$(F$(3),16,1):: N$=SEG$(N$
,1,4)&SEG$(VOR$,1,1):: RETUR
N
194Ø !
195Ø ! Input/Output-Routinen
196Ø !
197Ø INPUT #1,REC Z:F$(1),F$
(2),F$(3)
198Ø RETURN
199Ø INPUT #1,REC Z:F$(1),F$
(2),F$(3),F$(4),F$(5),F$(6),
F$(7),F$(8),F$(9),F$(1Ø),F$(
11),F$(12),F$(13)
2ØØØ RETURN
2Ø1Ø PRINT #1,REC Z:SEG$(F$(
1),1,1),SEG$(F$(2),1,1Ø),SEG
$(F$(3),1,27),SEG$(F$(4),1,2
7),SEG$(F$(5),1,7),
2Ø2Ø PRINT #1:SEG$(F$(6),1,2
Ø),SEG$(F$(7),1,15),SEG$(F$(
8),1,2Ø),SEG$(F$(9),1,27),SE
G$(F$(1Ø),1,27),
2Ø3Ø PRINT #1:SEG$(F$(11),1,
27),SEG$(F$(12),1,27),SEG$(F
$(13),1,5)
2Ø4Ø RETURN
2Ø5Ø PRINT #1,REC Ø:LSP,LP,B
US$,GEN$,DEV$,K$(1),K$(2),K$
(3),K$(4),K$(5),K$(6),K$(7),
K$(8),K$(9),K$(1Ø),NK,NL,PV$
2Ø6Ø RETURN
```



TI REVUE

Das Magazin für TI 99-4A

Nr. 7/87

So wird Ihr TI dreimal schneller

Fern-schreiber als Drucker

Window-Star: Bis zu zehn Fenster möglich

DAS HEFT IM HEFT FÜR TI-USER

EINFACH HERAUS-TRENNEN

VERALTETE E/A - ROUTINEN

In Ihrer Juli-87 Ausgabe sind nach langer Zeit 'mal wieder interessante Sachen zu lesen. Besonders gelungen fand ich den Beitrag von Reiner Ziegler und auch den Bericht über das GPL. Was mich aber sehr gestört hat, und weswegen ich diesen Brief schreibe, ist das Abdrucken der Liste der E/A-Routinen. Diese Liste ist im ersten und zweiten Assemblerbuches von Simon Koppelmann enthalten. Diese Bücher nennen wohl viele Assembler-Programmierer ihr Eigen. Der Arbeitsaufwand ist beim Erstellen der Liste gering, und die Idee und das Wissen kann den beiden Büchern entnommen werden. Diese Seiten hätte man wesentlich sinnvoller nutzen können.

Oliver Siffrin, Elversberg

Vielen Dank für Ihren Brief. Wir sind für Kritik immer offen, da wir so uns eher nach dem Leser richten können, als wenn eine schweigende Menge uns umgibt.

PROBLEME MIT TI-FORTH

Ich habe eine Freeware-Version von TI-Forth 2.0. Diese ist auf zwei Disketten, die einseitig und mit einfacher Dichte bespielt sind. Da ich jedoch zwei doppelseitige Laufwerke besitze, habe ich es auf eine Diskette kopiert. Lade ich nun Forth, so startet das System nicht mehr richtig. Der Verdacht liegt nahe, daß es nicht mehr ordnungsgemäß geladen wird. Welche Sysscreens muß ich ändern, damit alles wieder korrekt läuft?

H.P. Hamburg

Vielleicht ist ein Leser in der Lage, Auskunft zu geben.

DISKETTENSCHUTZ IN BASIC

Durch Schreiben eines P in Sektor 0 kann eine Diskette geschützt werden. Wie installiert und entfernt man diesen wieder?
Benötigt man dazu ein spezielles Programm, z.B. das DSK-Fixer Modul, oder geht dies auch mit einem kleinen Programm aus Assembler?
Wozu dient das Programm 'DYROFF' auf der Diskette von TI-REVUE 4/87?
Für das DIS-ASS Programm aus TI-REVUE 1/86 benötigt man noch einen Disassembler. Woher kann man einen solchen bekommen, da alle Disassembler, die ich besitze, ungeeignet sind?

J.S. München

Das P befindet sich in Sektor 0 auf Byte 16. Ändern können Sie dies mit jedem Programm, das Sektoren editiert. Ein kleines Assembler-Programm, das dieses Byte ändert, geht auch. Dazu lesen Sie den Sektor, verändern Ihn und schreiben ihn anschließend wieder.
Das Programm 'DYROFF' ist das GPL-Programm, das Herr Dyroff in seinem GPL-Kurs bespricht. Es ist als File für die GRAM-Karte abgespeichert.
Der Disassembler, für den das Programm 'DIS-ASS' ausgelegt ist, ist der Disassembler von Michael Weiand. Sie können aber jeden beliebigen dazu verwenden. Das Ergebnis muß nur eine Dis/Fix 80 Datei sein, und der Quellcode an Tabulatorposition 15 beginnen.

TI-DISKETTENSYSTEM UNKLAR

Ich habe den Heimcomputer TI-99 mit einem verhältnismäßig großem Ausbau. Obwohl ich gehörlos bin, verwende ich gerne den TI. Leider klappt mit dem Diskettensystem etwas nicht. Die Diskette will mir ihren Inhalt nicht anvertrauen. Kann es sein, daß das Diskmanager-Modul dabei eine wichtige Rolle spielt?

Heinz Goldbaum, Flensburg

Ein Diskmanager ist notwendig, den Diskettenkatalog aufzulisten, Disketten zu formatieren, etc. Ohne einen Diskmanager läuft also nichts. Der Diskettenkatalog kann aber auch noch als Datei eingeladen werden. Beziehen können Sie einen Diskmanager von T.I-Clubs, die solche Programme als Freeware besitzen.

TASTATUR PRELLT PLÖTZLICH

Seit einiger Zeit hat meine Tastatur angefangen, auf einigen Tasten zu prellen. Manchmal hilft nicht einmal mehr ein ganz kurzes Antippen, wenn man nicht gleich eine ganze Serie von Buchstaben auf dem Bildschirm haben will. Ist meine Tastatur durch den häufigen Gebrauch dem Verschleiß unterlegen?

Paul Müller, Ingolstadt

Das Prellen Ihrer Tastatur ist, wenn Sie sie immer sanft behandelt haben und noch ein altes Modell besitzen, keinesfalls eine Verschleißerscheinung. Die alten Tastaturen sind

IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Geschäftsführer: Werner E. Seibt
Redaktion: Alfons Bauer, Thomas Dehn, Jürgen Puschmann, Uwe Schröder
Verantwortlich für den Inhalt:
MSX-Teil: Werner E. Seibt
TI-Teil: Oliver Huber (Technik) Hartmut König (Listings)
Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d
Alle: Postfach 1161
8044 Lohhof.
Anfragen bitte nur schriftlich
Druck: Dierichs, Kassel
Es gilt die Honorarliste des Verlages.
Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.
Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

©by CA Verlag, Heßstraße 90, 8000 München 40
SPS und Autoren

noch, im Gegensatz zu den neuen, an den Kontakten offen. Es kann also Staub eindringen. Das Problem läßt sich ganz einfach beheben. Die Tastenkappen lassen sich senkrecht nach oben von der Tastatur abziehen. Den Staub entfernen Sie z.B. mit einer Preßluftdose, mit der man auch Photoapparate reinigt. Danach müßte alles wieder funktionieren. Haben Sie allerdings eine neue Tastatur, so liegt wahrscheinlich ein Defekt vor.

TI99/4A

PREISSENKUNG BEI VIELEN ARTIKELN !!

Peripherie

Computer TI-99/4 A komplett	249,–
Platine TI-99/4 A	59,–
Tastatur TI-99/4 A	39,–
32 K-Karte (Atronic)	249,–
RS 232-Karte (Atronic)	249,–
Disccontroller DSDD (Corcomp)	399.–
P-Code-Karte (gebraucht)	399,–
80-Zeichen-Karte (Mechatronic)	449,–
128 K GRAM-Karte (Mechatronic)	499,–
CPS 99 mit 1 Diskettenlaufwer DSDD + Statistikmodul	999,–
CPS 99 mit 2 Diskettenlaufwerken DSDD + Statistikmodul	1299,–
Diskettenlaufwerk DSDD mit Einbausatz für Peripheriebox	329,–
2 Diskettenlaufwerke DSDD mit Einbau- und Kabelsatz für Peripheriebox	679,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle	249,–
Ext. 128 K-Erweiterung mit Centronicsschnittstelle + Druckerkabel + Epsondrucker LX 800	869,–
dto. + Epsondrucker FX 800	1299,–
dto.+ Commodoredrucker MPS 1000	959,–
dto.+ Stardrucker NL-10	899,–
RGB-Modulator mit Scartstecker	179,–
RGB-Modulator + Farbmonitor 1081	869,–
MBX-Sprachsteuereinheit + Baseball + Sewermania anschlußfertig	249,–
Grafiktablett Supersketch	99,–
Supersketch + Defender + Dig Dug + Donkey Kong+ Statistik	149,–
WIPO Mouse 99 anschlußfertig	179,–
Akustikkoppler Dataphon S21d + Kabel für RS 232 Karte	279,–
Modulexpander 3-fach	125,–
Fernbedienung (Original TI)	49,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter)	115,–

Software

Extended Basic (Original TI)	129,–
Extended Basic II Plus	229,–
Mini Memory	149,–
Mini Memory+ Assemblerbuch dt.	199,–
Editor/Assembler (32 K + Disc notw.)	149,–
TI-Logo II (32 K + Disc notwendig)	179,–
Diskfixermodul (Navarone)	99,–
Navarone Database monagement	99,–

Terminal Emulator II	59,–
TI-99/4 A Flyer (Flugsimulation)	69,–
Attack, Chisholm Trail, Amazing	je 19,–
Car Wars, Moon Mine, Munch Man, Invaders. Tombstone, Statistik, Defender, Dig Dug	je 29,–
Espial, Fathom, Soccer, Parsec, Treasure Island, Jungle Hunt, Moon Patrol, Shamus, Donkey Kong	je 39,–
Adventuremodul, Buck, Rogers, Burger time, Congo Bongo, Microsurgeon, Moonsweeper, Popeye, Tunnels of doom, Pirates Isle, Pole Position	je 49,–
Video Chess (Schachmodul)	59,–
Dig Dug+ Moon Patrol	49,–
Defender+ Moon Mine	49,–
Parsec+ Shamus	59,–
Espial + Moonsweeper	69,–
Fathom + Burgertime	69,–
Buck Rogers+ Jungle Hunt	69,–
Defender + Dig Dug + Jungle Hunt + Shamus	99,–
TI-Artist (Diskette)	99,–
Turbo Pascal (Diskette)	149,–
Window Manager für Turbo Pascal	99,–
C'99 Compiler (Diskette)	99,–

Komplettübersicht über weiteres Zubehör + Modulpakete in unserer Preisliste !!!!!

Bücher

Anwender-Handbuch Turbo Pascal dt.	39,–
Editor/Assembler Handbuch dt.	98,–
TMS 9900 Assemblerhandbuch für das Mini Memory deutsch	78.–
TI-99/4 A Intern dt.	38,–
Assembler-Kurs (99er-Journal)	59,–
Introduction Assembly Language	29,–

Commodore/Schneider

Commodore C128 D	1049,–
Amiga 500+ Farbmonitor 1081	1769,–
Amiga 2000 + Farbmonitor 1081	3079,–
Commodore Plus 4 + Floppy VC 1551	469,–
Schneider CPC 6128 Grünmonitor	729,–
CPC 6128 Farbmonitor	1149,–
PC 1640/1 Laufwerk/SW-Monitor	1549,–
2 Laufwerke/SW-Monitor	1979,–
1 Laufwerk/Farbmonitor	1979,–
2Laufwerke/Farbmonitor	2429,–

Versandkostenpauschale (Warenwert bis DM 1000,–/darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–). Lieferung nur gegen NN oder Vorauskasse: Ausland nur Vorauskasse. Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstr. 5, 7324 Rechberghausen, Tel. (0 71 61) 5 28 89

FERNSCHREIBER ALS DRUCKER AM TI 99/4A

Konfiguration: Konsole, MINIMEM, (oder EXT-BASIC mit Speichererweiterung und Programmumstellung), Fernschreiber, Schnittstelle (Eigenbau) nebst Netzteil (45 - 60 V, 60 mA; 5 V 100 mA).

Das für MINEMEM ausgelegte Programm belegt den Bereich >7E00 – >7FD0 d.h. 464 Byte. Für die Speicherung einer Bildschirmzeile wird zusätzlich der Bereich >7DEO – >7DFF genutzt.

ZUR HARDWARE:

Es wurde die Dekodierschaltung des in der TI-Revue veröffentlichten E-Promers verwendet, CRU Adresse = >1910. Die für den Betrieb des Fernschreibers aufbereiteten seriellen Daten werden an PIN 4 des 74LS 259 abgenommen und über den Vorverstärker dem Optokoppler zugeführt. Mit der Sekundärseite werden 45 Volt DC über eine einstellbare Strombegrenzung (10-50 mA) gemäß Skizze (Belegung des Fschr. Steckers) dem Fschr. zugeschaltet.

Nun eine kleine Erinnerung an die Funktion der Innereien des Fschr: Es werden seriell 5 BIT, synchron zu 50 Hz nach einem Startbit,dem Fschr. zugeführt. Jedes BIT wird mechanisch einem von 5 Anzugsmagneten zugeordnet (HIGH oder LOW; Anzug oder nicht). Jedes BIT muß 20 msec lang sein (bei 50 BAUD). Bei einem Spulenwiderstand von 200Ω und einem vorgeschriebenen Strom von 20 mA erhebt sich die Frage: wozu wird die hohe Spannung benötigt? Beim Anlegen einer Spannung an eine Induktivität muß von der Spannung in der Induktivität ein Magnetfeld aufgebaut werden. Für den Zeitraum des Aufbaus des Magnetfeldes addiert sich zum Ohmschen Widerstand der Spule ein induktiver Widerstand. Mit Hilfe eines Oszillographen (Anschluß am Fschr.Stecker) läßt sich diese Erklärung nachprüfen.

Bis 0,2 mSec nach dem Aktivieren eines Magneten beträgt die Spannung an der Spule mehr als 25 V und erst nach mehr als 4 mSec 4 Volt (Strombegrenzung 20 mA). D.h., erst nach 4 mSec wirkt nur noch der Ohmsche Anteil der Spule. Nach 0,2 mSec ergibt sich der Gesamtwiderstand der Spule mit 25 V/0,02 mA = 1250 Ω, somit beträgt der induktive Anteil 1050Ω.

Die hohe Spannung ist notwendig, um einen möglichst schnellen Aufbau des Magnetfeldes zu erreichen, ohne daß im eingeschwungenen Zustand der Dauerstrom zu groß wird.

ZUR SOFTWARE:

Es sind zwei Programme vorhanden

1) CALL LINK ("TASTEN")

Bitte lesen Sie weiter auf Seite 36!

```
                     ASCI-CODE          ASCI-ZEICHEN

7E00  457A        DATA  >457A   * E  z  (klein "z" = Zeilenvorschub)
   2  4120        DATA  >4120   * A
   4  5349        DATA  >5349   * S  i
   6  5572        DATA  >5572   * U  r  (klein "r" = Rücklauf)
   8  4452        DATA  >4452   * D  R
   A  7A4E        DATA  >4A4E   * J  N
   C  4643        DATA  >4643   * F  C
   E  4B54        DATA  >4B54   * K  T
7E10  5A4C        DATA  >5A4C   * Z  L
   2  5748        DATA  >5748   * W  H
   4  5950        DATA  >5950   * Y  P
   6  514F        DATA  >514F   * Q  O
   8  4247        DATA  >4247   * B  G
   A  7C4D        DATA  >7C4D   *    M  >7C ≙ Ziffernumschaltung
   C  5856        DATA  >5856   * X  V
   E  7B7F        DATA  >7B7F   *       >7B ≙ Buchstabenumschaltung
7E20  337A        DATA  >337A   * 3  z
   2  2D20        DATA  >2D20   * -
   4  2238        DATA  >2238   * "  8
   6  3772        DATA  >3772   * 7  r
   8  2634        DATA  >2634   * &  4    ✣ ≙ &
   A  402C        DATA  >402C   * a  ,    ◒ ≙ a
   C  3E3A        DATA  >3E3A   * `  :    ▫ ≙ >
   E  2835        DATA  >2835   * C  5
7E30  2B29        DATA  >2B29   * +  )
   2  3224        DATA  >3224   * 2  $    ⍁ ≙ $
   4  3630        DATA  >3630   * 6  0
   6  3139        DATA  >3139   * 1  9
   8  3F2A        DATA  >3F2A   * ?  *    ⊟ ≙ *
   A  2C2E        DATA  >2C2E   *    .
   C  2F3D        DATA  >2F3D   * /  =
   E  7B7F        DATA  >7B7F
7E40  2000   EY   DATA  >2000
   2  1F00   P1   DATA  >1F00
   4  0040   P2   DATA  >0040
   6  001F   P3   DATA  >001F
   8  1B00   P4   DATA  >1B00
   A  0200   P5   DATA  >0200
   C  0800   P6   DATA  >0800
   E  1000   P7   DATA  >1000
7E50  0200   ES   LI     R0, >7F30       * Einsprung "SCHIRM" SPRUNG
   2  7F30                                 zum Unterprogramm (PS) vor-
   4  C800        MOV    R0, a >7E92       bereiten
   6  7E92
   8  0200        LI     R0, >106C
   A  106C
   C  C800        MOV    R0, a >7ECC
   E  7ECC
7E60  100C        JMP    BG
   2  1000        NOP
   4  1000        NOP
   6  0200   ET   LI     R0, >7F84       *  Einsprung "TASTEN" Sprung
   8  7F84                                  zum Unterprogramm (AY) vor-
   A  C800        MOV    R0, a >7E92        bereiten
   C  7E92
```

```
E     0200      LI    R0,     >10E1       * Sprungadresse für Progr.
7E70  10E1                                  "TASTEN"
2     C800      MOV   R0, a >7ECC
4     7ECC
6     1000      NOP
8     1000      NOP
A     04C6  BG  CLR   R6
C     020C      LI    R12, >1910          * CRU-Adresse
E     1910
7E80  020D      LI    R13, >0D00          * Verzögerungswert für
2     0D00                                  Stopbitschleife
4     020E      LI    R14, >06FA          * Verzögerungswert für
6     06FA                                  Datenbitschleife
8     C220      MOV   a >837C, R8         * STATUS-sichern
A     837C
C     C28B      MOV   R11, R10            * Rücksprung sichern
E     1000      NOP
7E90  06A0  AN  BL    a >7F30             * Adr PS oder Ay
2     7F30
4     0586      INC   R6                  * Zeichenzahl +1
6     81A0      C     a P3, R6            * schon 31 Zeichen ?
8     7E46
A     160A      JNE   F3                  * nein dann F3
C     C160      MOV   a >7E4A, R5         * Code P5 für Rücklauf
E     7E4A                                  laden
7EA0  06A0      BL    a SG                * Rücklauf durchführen
2     7FAE
4     C160      MOV   a P6, R5            * Code P6 für Zeilenvor-
6     7E4C                                  schub laden
8     06A0      BL    a SG                * Zeilenvorschub durch-
A     7FAE                                  führen
C     04C6      CLR   R6                  * Zeichenzahl auf 1 setzen
E     0586      INC   R6
7EB0  0203  F3  LI    R3, >7A00           * ASCI für klein z laden
2     7A00
4     9043      CB    R3, R1              * ist "z" gefordert ?
6     160D      JNE   F4
8     C160  F5  MOV   A P5, R5            * Rücklauf laden
A     7E4A
C     06A0      BL    a SG                * und durchführen
E     7FAE
7EC0  C160      MOV   a P6, R5            * Zeilenvorschub laden
2     7E4C
4     06A0      BL    a SG                * durchführen
6     7FAE
8     04C6      CLR   R6                  * Zeichenzähler auf 0
A     1000      NOP
C     106C      JMP   >106C               * AN oder CK ("TASTEN" oder
E     1000      NOP                         "schirm")
7ED0  1000      NOP
2     04C4  F4  CLR   R4                  * Abfragezähler = 0
4     0205      LI    R5, >7E00           * ASCI-Listenanfang
6     7E00
8     D1D5  F2  MOVB  *R5, R7             * Erstes ASCI-Zeichen holen
A     0584      INC   R4                  * Abfragezähler +1
C     8120      C     a P2, R4            * schon 63 Abfragen ?
E     7E44
```

```
7EE0   1307          JEQ    F
   2   91C1          CB     R1, R7         * ist Zeichen aus Liste = R1?
   4   1603          JNE    F1             * nein dann nächsten Listenwert
   6   06A0          BL     a TR           * ja; Fschrstand prüfen und ggf.
   8   7EFE                                  korrigieren
   A   10D2          JMP    AN
   C   0585    F1    INC    R5             * ASCI-Listen-adresse +1
   E   10F4          JMP    F2             * nächstes Zeichen aus Liste
7EF0   0205    F     LI     R5, >0400      * Fschr.code für Blank laden
   2   0400
   4   06A0          BL     a SG           * Ausgabe Blank wenn unbek. Zeich.
   6   7FA0
   8   10CB          JMP    AN             * neues Zeichen oder Zeile holen
   A   1000          NOP
   C   1000          NOP
   E   C00B    TR    MOV    R11, R0        * Rücksprung sichern
7F00   SWPB          SWPB   R4
   2   9804          CB     R4, a >7E42    * schon 31 Abfragen?
   4   7E42
   6   140C          JHE    Z1
   8   9809          CB     R9, a >7E42    * steht Fschr auf Buchstaben-
   A   7E42                                  mod.? (31)
   C   1305          JEQ    Z2
   E   C260          MOV    a P1, R9       * nein, Buchstabenmod nach R9
7F10   7E42
   2   C149    Z3    MOV    R9, R5         * nach R5
   4   06A0          BL     a SG           * umschalten in Buchstabenmode
   6   7FAE
   8   C144    Z2    MOV    R4, R5         * Zeichen nach R5 ausgeben
   A   06A0          BL     a SG
   C   7FAE
   E   0450          B      * R0           * Rücksprung
7F20   9809    Z1    CB     R9, a >7E48    * Fschr.stellung = P4
   2   7E48
   4   13F9          JEQ    Z2             * ja; zur Zeichenausg.
   6   C260          MOV    a P4, R9       * umsch. Zeichen nach R9
   8   7E48
   A   10F3          JMP    Z3             * zur Umschaltung
   C   1000          NOP
   E   1000          NOP
7F30   0200    PS    LI     R0, >2C0       * Adresse vorletzte B-Zeile
   2   02C0
   4   0201          LI     R1, >DE0       * Adresse-Speicher
   6   7DE0
   8   0202          LI     R2, >0020      * Zahl der Zeichen
   A   0020
   C   0420          BLWP   a >6030        * Zeichen vom Schirm holen
   E   6030
7F40   C801          MOV    R1, a P7       * Anfangsadr sichern (spalten)
   2   7E4E
   4   0200          LI     R0, >7F4C      * neue Sprungadresse
   6   7F4C
   8   C800          MOV    R0, a >7E92    * zum Segment "AN"
   A   7E92
   C   0200    PT    LI     R0, >7E00      * Letzte Speicher pos. +1
   E   7E00
```

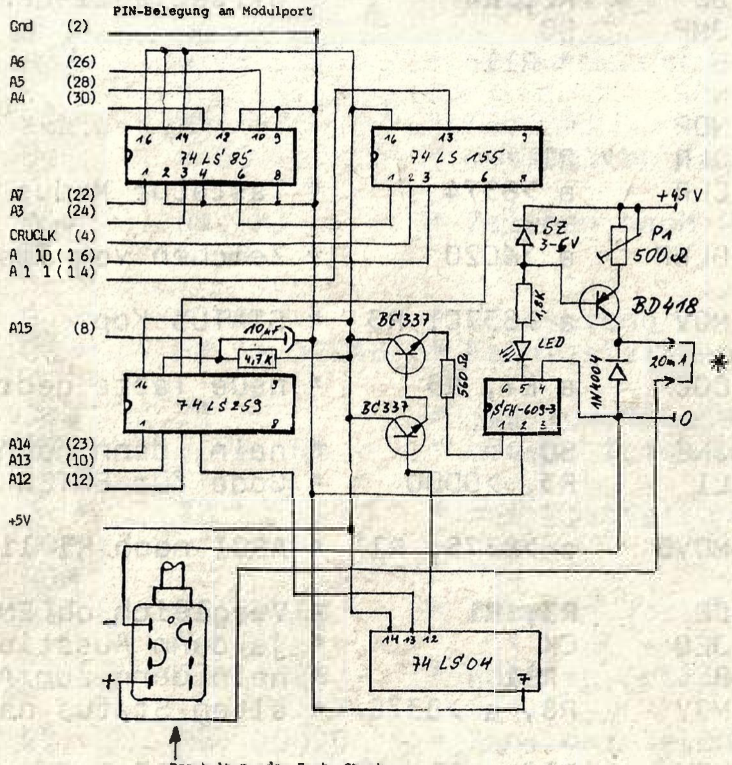
```
7F50  0202       LI     R2, >6000      * offset im linken Byt
   2  6000
   4  C0E0       MOV    a >7E4E, R3    * Spaltenadresse
   6  7E4E
   8  04C1       CLR    R1
   A  0053       MOVB   *R3, R1        * Zeichen von Adresse in R3 nach
   C  0583       INC    R3           R  1
   E  80C0       C      R0, R3         * Vergleich ; Zeilenende ?
7F60  13AB       JEQ    F5             * ja; nach F5 (Rücklauf + z)
   2  C803       MOV    R3, a >7E4E    * nein: Adresse von R3 nach 7E4E
   4  7E4E
   6  7042       SB     R2, R1         * offset abziehen
   8  C101  PU   MOV    R1, R4
   A  0207       LI     R7, >2000      * ASCI-Blank
   C  2000
   E  9107  B0   CB     R7, R4
7F70  1106       JLT    B1             * JUMP IF R7 kleiner als R4
   2  0583       INC    R3             * Speicheradr. +1
   4  80C0       C      R0, R3         * Vergleich: Speicherende
   6  13A0       JEQ    F5             * Ja: Rücklauf und Zeilenvor.
   8  D113       MOVB   *R3, R4        * nein: neues Zeichen aus Speicher
   A  7102       SB     R2, R4         * offset abziehen
   C  10F8       JMP    B0
   E  045B  B1   B      * R11
7F80  1000       NOP
   2  1000       NOP
   4  04C1  Ay   CLR    R1
   6  04E0  SC   CLR    a >8374        * Tastatur Modus 0
   8  8374
   A  0420       BLWP   a >6020        * Zeichen von Tastatur
   C  6020
   E  C0E0       MOV    a >837C, R3    * STATUS Kop
7F90  837C
   2  20E0       COC    a EY, R3       * neue Taste gedrückt
   4  7E40
   6  16F7       JNE    SC             * nein, dann nochmal
   8  0203       LI     R3, >0D00      * Code für ENTER
   A  0D00
   C  0060       MOVB   a >8375, R1    * ASCI nach R1 linkes Byte
   E  8375
7FA0  9043       CB     R3, R1         * Vergleich ob ENTER
   2  1301       JEQ    CK             * ja dann Ausstieg
   4  045B       B      *R11           * nein dann zum Aufruf
   6  C808  CK   MOV    R8, a >837C    * alten Status nach 837C
   8  837C
   A  C2CA       MOV    R10, R11       * alten Rücksprung
   C  045B       B      * R11          * Rücksprung
   E  0545  SG   INV    R5
7FB0  0A15       SLA    R5, 1          * Zeichen um 1 BIT verlängern
   2  0202       LI     R2, 6
   4  0006
   6  3045       LDCR   R5, 1          * 1 BIT ausgeben
   8  06C5       SWPB   R5             * Byte tauschen
   A  0815       SRA    R5, 1          * um 1 BIT verkleinern
   C  06C5       SWPB   R5             * Byte tauschen
   E  C0CE       MOV    R14, R3
7FC0  0603       DEC    R3               Verzögerung 20 msec
```

```
   2     16FE      JNE       >7FC0
   4     0602      DEC       R2              * nächste BIT holen
   6     16F7      JNE       >7FB6
   8     3043      LDCR      R3, 1           * Stopbit ausgeben
   A     C0CD      MOV       R13, R3
   C     0603      DEC       R3                Verzögerung 40 msec
   E     16FE      JNE       >7FCC
7FD0     045B      B         *R11

             701C      7FD2                  * Erste freie Adresse
             701E      7FE0                  * Beginn Namensliste

             7FE0      5343                  * SC
                2      4849                  * HI
                4      524D                  * RM
                6      7E50                  * Programmstart
                8      5441                  * TA
                A      5354                  * ST
                C      454E                  * EN
                E      7E66                  * Programmstart
```

Nach dem Aufruf (aus BASIC) wird jeder Tastenwert mit dem Fschr. geschrieben: nach 32 Zeichen erfolgt Rücklauf und Zeilenvorschub. Ausstieg mit Enter.

2) CALL LINK ("SCHIRM")
Nach dem Aufruf wird die vorletzte Bildschirmzeile in den Speicherbereich>7DE0 – > 7DFF übernommen, dann Zeichen für Zeichen dem Aufbereitungsprogramm zugeführt und vom Fernschreiber ausgedruckt. Die Adresse für die Bildschirmzeile liegt auf >7F32 und > 7F33 (hier >2C0). Durch Änderung mit CALL LOAD kann auch jede andere Zeile abgerufen und ausgedruckt werden. Nach jedem gedruckten Zeichen wird geprüft (PU), ob der Rest der Zeile nur noch "BLANK" enthält. Wenn ja, wird Wagenrücklauf sowie Zeilenvorschub durchgeführt und dann das Programm beendet (CK).

PIN-Belegung am Modulport



Beschaltung des Fschr.Steckers

* Einstellung auf 20 mA (evtl. 40 mA; Typenabhängig) mit P 1.

Verwendete Bauelemente unkritisch.

ALLGEMEINE BESCHREIBUNG:

In dem Bereich > 7E00 – > 7E3F liegt der ASCII-CODE der erlaubten Zeichen in der Reihenfolge, die dem Fernschreibcode (CCITT Nr. 2) entspricht. Beispiel: Der Buchstabe K ist durch seinen ASCII-CODE (4) an fünfzehnter Stelle (> 7E0E) plaziert. Wenn also die Ausgabe des Buchstabens K gefordert wird, passiert folgendes:

Es wird von > 7E00 bis > 7E3F nacheinander jedes Byte mit dem Inhalt von Register 1 verglichen (und mit jedem Vergleich der Inhalt von R 4 um 1 erhöht) bis der Vergleich positiv ist. In unserem Beispiel steht hier dann in R4 die Ziffer 15. Bei Ausgabe einer 15 (seriell, IIII0) zum Fernschreiber schreibt dieser ein K, wenn er im Buchstabenmode steht.

UMSCHALTEN DES FERNSCHREIBERS:

Da nur 5 BIT zur Übertragung zur Verfügung stehen (somit max. 32 Zeichen) aber 52 Zeichen plus Umschaltbefehle möglich sind, ist eine Umschaltung des Fernschreibers nach folgendem Verfahren notwendig:

ERSTES BEISPIEL:

Buchstabe "K" steht an fünfzehnter Stelle, d.h. der Fschr. muß im Buch-

stabenmode stehen (Stand wird im Register 9 dokumentiert), steht im Register 9 nicht 31 (=Umschaltcode in Buchstabenmod.), wird 31 nach R9 geschrieben und zum Fernschr. ausgegeben. Danach erfolgt die Ausgabe von R5 = 15.

ZWEITES BEISPIEL:

gefordert" ("Klammer auf. ASCI-Code> 28 steht an Stelle 47. Da der Wert größer 32 ist, muß in R9 der Umschaltcode auf Zeichenmode stehen. Falls nicht, wird 27 =>1B nach R9 geschrieben und zum Fernschreiber ausgegeben. Danach erfolgt die Ausgabe von R5 = (47-32) = 15.

Je nach Stellung des Fernschreibers wird bei Ausgabe von 15 = IIII0 oder 47 = IIII0I (das sechste BIT wird ignoriert) ein "K" oder "("ausgegeben. Sollte ein angefordertes Zeichen mit seinem ASCI-Code nicht im Speicher > 7E00 — > 7EF0 enthalten sein, wird BLANK ausgegeben.

Für Programmerweiterungen wurden zwischen den Programmteilen Leerstellen (NOP- 10000) vorgesehen; sie erleichtern das Einfügen von Sprungbefehlen.

SONDERZEICHEN

Die am Fernschreiber vorhandenen Sonderzeichen werden wie folgt zugeordnet:

□≙> Ω≙@ ⊠≙$ ⊟≙*

Die folgende Kombination kann nur nach einem mechanischen Eingriff in den Fernschreiber (Unterdrückung der Kennungsausgabe) genützt werden:

✚≙&

UNTERPROGRAMM-BESCHREIBUNGEN

ES Einsprung für Programm "SCHIRM" Vorbereitung durch Setzen der Sprungadressen

ET Einsprung für Programm "TASTEN" Vorbereitung durch Setzen der Sprungadressen

BG Register setzen, Status und Rücksprungadresse sichern

AN Verzweigung zu PS ("SCHIRM") oder AY ("TASTEN")

PS Die vorletzte Bildschirmzeile wird in den Bereich 7DE0 bis 7DFE gespeichert

PT Einzelabtastung aus Speicher aus > 7De0 aufsteigend

PU Unterprogramm zur Prüfung des Zeilenrestes. Wenn nur noch Blank, dann Zeilenvorschub und Wagenrücklauf einleiten. Rücksprung zu AN+2

AN+2 Umsetzen ASCI in Fschrcode; unbekanntes Zeichen als Blank ausgeben;

TR Fschrzustandskontrolle (Zeichen oder Buchst.) in R9 richtigen Zustand herstellen und Fschr. gegebenenfalls umschalten

F3 Prüfen, ob über ASCI "Z" Zeilenvorschub und Rücklauf gefordert; gegebenenfalls durchführen

SG Zeichenausgabe, Zeichen um Startbit verlängern und BIT für BIT ausgeben. Zwischen den Ausgaben 20 msec Schleife. Stopbit ausgeben (40msec)

AY Einsprung "TASTEN"; TASTATUR Abfrage und Ausgabe; Ausstieg mit Enter

F2 ASCII-Code aus Speicher holen. Prüfen, ob schon 63 Vergleiche durchgeführt. Vergleich durchführen.

EINFÜHRUNG IN GPL (VII)

Dies ist die vorletzte Folge dieser Serie und wahrscheinlich auch mit dem letzten Demoprogramm. Es enthält nur zwei neue Befehle, die aber sind in ihrer Anwendung äußerst effektiv und erleichtern das Programmieren sehr. Es sind außerdem die letzten wichtigen GPL-Anweisungen, die Sie kennen müssen, um gute Programme schreiben zu können. Wer weiß, vielleicht steht Ihr Copyright demnächst unter einem GPL-Listing in der TI-Revue!?

Wie wichtig Stackoperationen sein können, wissen Sie bereits von den Befehlen CALL und RTN her. Dort wird die jeweilige Rücksprungadresse auf einen Stack gelegt und später wieder geholt. Das, was zuletzt geschrieben wurde, wird zuerst wieder gelesen. Es

PUSH: STACKOPERATIONEN SIND MEHR ALS NUR SPIELEREI

muß dazu kein spezieller Bereich für irgendwelche Tabellen reserviert werden. Der Stack wächst halt einfach immer weiter und schrumpft beim Lesen wieder zusammen. Einige von Ihnen haben bestimmt auch schon mit dem Ed/Ass oder dem Minimem gearbeitet und wissen, daß beim Laden mehrerer Maschinenprogramme deren Namen ebenfalls auf einen Stack kommen, nämlich die sogenannte REF/DEF-Table. Sie kann fast beliebig lang werden, nimmt aber nie mehr Platz weg als nötig. Überhaupt sind Stacks sowohl in der Datenverarbeitung und -verwaltung wie auch bei Subroutinen und mathematischen Operationen sehr beliebt geworden, daher unterstützt auch GPL diese Funktionen. PUSH besitzt nur einen Operanden. Diesen Wert legt PUSH auf den GPL-Datastack, nicht zu verwechseln mit dem GPL-Substack für die Rücksprungadressen! Auf den Datastack können Sie alle möglichen Werte legen, aber immer nur byteweise! Es gibt zwar auch einen DPUSH, aber das Hi-Byte des Wertes wird von ihm ignoriert; auch er legt aufgrund einer unvollständigen Dekodierung im Interpreter nur Bytes auf den Stack! Das Lesen dieser Stackdaten geschieht ▶▶

nicht mit einem Befehl, sondern mit einem Operanden: * > 837C (siehe letzten Bericht über Sonderformate bei den Operanden). ST § > 8300, * > 8300C z.B. kopiert den obersten Wert vom Stack nach § > 8300 und erniedrigt den Stackpointer um eins. Auch beim Lesen gibt es eine Besonderheit bei Wortoperationen: DST § > 8300, * > 837C holt sich zwar die obersten ZWEI Bytes vom Datastack und kopiert sie nach § > 8300 und § > 8302, der Stackpointer hingegen wird aber nur um EINS erniedrigt! BEIM NÄCHSTEN STACKZUGRIFF WÜRDE DAS 2. BYTE ALSO NOCHMAL GELESEN! Deshalb führen Sie Stackoperationen in GPL bitte immer nur byteweise durch. Sie ersparen sich damit u.U. sehr viel Ärger.

COINC: Leider auch noch nicht ganz enträtselt.

Sie kennen CALL COINC vom TI-Extended-Basic her. Sie wissen, daß Sie damit das Zusammentreffen zweier Sprites oder eines Sprite mit einer bestimmten Bildschirmposition unter Berücksichtigung gewisser Toleranzen feststellen können. Genau dies tut COINC in GPL auch. Ein Blick in das Handbuch zum E/A sagt Ihnen, daß es im VDP-RAM eine gewisse Spritewertetabelle gibt, in der zu jedem Sprite 4 Bytes gehören: Y-Pos, X-Pos, Char und Status/Farbe.

Zu COINC gehören zwei Operanden. Die zugehörigen Werte müssen WORTE sein! Das Hi-Byte ist eine Y-Pos, das Low-Byte eine X-Pos. COINC selbst arbeitet nämlich gar nicht unbedingt nur mit Spritedaten. COINC errechnet nur, ob sich die durch die beiden Operanden beschriebenen Koordinaten noch innerhalb eines gewissen Abstandes zueinander befinden. Sagen wir, die Spritewertetabelle beginnt bei VDP§ 0800. Dann errechnet COINC VDP§ > 0810, VDP§ > 0814 eine eventuelle Coincidence zwischen den Positionen der Sprites Nr. 5 und 6. Und COINC VDP§ > 0800 >1020 prüft, ob sich der Sprite Nr. 1 auf der Position (16/32) befindet (Im Grunde müßte man eigentlich immer DCOINC schreiben. Dies ist aber nicht notwendig, da COINC als Byteoperation wie gesagt wenig sinnvoll ist, es sei denn, die Y-Koordinaten der Sprites wären ständig Null). Zur Angabe der Toleranz stehen nach COINC noch drei Bytes, und hier beginnt auch das Dilemma: Was diese Bytes nämlich sollen, ist mir nicht ganz klar. Da ist zunächst ei-

FINDEN SIE DOCH MAL DIE LÖSUNG!

nes, das im ROM-Listing von Heiner Martin als "Mapping" bezeichnet wird. Die Differenzen der Y- und X-Angaben beider Operanden von COINC werden so oft durch zwei geteilt, wie es dieses Byte angibt. Was das soll – fragen Sie mich bitte nicht. Anschließend stehen zwei Bytes, die auf eine Tabelle im GROM zeigen müssen. Diese Tabelle enthält offensichtlich in irgendeiner Form die Angaben über die zulässigen Toleranzen. Aber wie und in welcher Form war nicht herauszubringen. Vielleicht finden Sie ja die Lösung? Ich kann Ihnen jedenfalls vorläufig nicht mehr sagen und im Demoprogramm kommt COINC deshalb auch nicht vor. Hier nochmal sein prinzipielles Format:

COINC 1. Operand, 2. Operand
BYTE Mapping-Byte
DATA GROM-Tabellenadresse

HAT EINEN EIGENEN INTERPRETER: FMT

Bisher haben wir Graphiken und Texte mit MOVE und ST auf den Bildschirm gebracht. Es geht aber noch viel, viel einfacher. FMT ruft einen Subinterpreter in GPL auf, der nur eine einzige Aufgabe hat: Mit insgesamt 14 Befehlen die Bildschirmverwaltung zu übernehmen. Wichtig: Die Bezeichnung der Befehle stimmt nicht mit der Symbolik im ROM-Listing überein. Es sind vielmehr diejenigen Symbole verwendet, mit denen der GPL-Assembler von Michael Wieand arbeitet. Die Symbole aus dem ROM-Listing sind statt dessen jeweils in der 2. Zeile angegeben.

HTEX 'String'
. . . ': String:'
Ein String wird auf den Bildschirm horizontal, also waagrecht, geprinted. Er darf max. 32 Bytes lang sein. Das Byte für diesen Befehl errechnet sich aus der Stringlänge plus >FF:
Z.B. für HTEX 'HALLO' bzw. . . . ':HALLO:' schreiben Sie 04 48 41 4C 4C 4F (hex).

VTEX 'String'
ROM-Listing: nicht vorhanden
Ein String wird auf den Bildschirm vertikal, also senkrecht, geprinted. Er darf max. 32 Bytes lang sein. Das Byte für diesen Befehl errechnet sich aus der Stringlänge plus >1F:
Z.B. für VTEX 'HALLO' schreiben Sie 24 48 41 4C 4C 4F (hex).

HCHA Anzahl, 'Char'
. . . Anzahl ':Char:'
Ein Zeichen wird auf den Bildschirm horizontal max. 32-mal geschrieben (wie CALL HCHAR im Basic). Das Byte für diesen Befehl errechnet sich aus der Anzahl plus >3F:
Z.B. für HCHA 18, "*" bzw. . . . 18 ':*:' schreiben Sie 51 2A (hex).

VCHA Anzahl, 'Char'
ROM-Listing: nicht vorhanden
Ein Zeichen wird auf den Bildschirm vertikal max. 32-mal geschrieben (wie CALL VCHAR im Basic). Das Byte für diesen Befehl errechnet sich aus der Anzahl plus >5F:
Z.B. für VCHA 18, "*" schreiben Sie 71 2A (hex).

COL+ Anzahl
. . . Anzahl <
Bisher haben wir noch kein Wort darüber verloren, auf welche Stellen am Bildschirm eigentlich geschrieben wird. Abhängen tut es von den Speicherstellen > 837E (YPT) und > 837F (YPT), die bereits in den GPL-Berichten III (siehe beim Befehl ALL) und VI (siehe "Zugriffe auf >837D") erwähnt wurden. Beim Aufruf des Subinterpreters mit FMT verändert der Computer sie nicht, sie werden aber beim Printen immer mitgezählt und geben somit ständig die Position an, auf die das nächste Zeichen kommt. COL+ verschiebt diese Position in horizontaler Richtung um max. 32 Stellen nach rechts weiter. Bei Überschreitung des rechten Bildrandes geht es links eine Zeile tiefer weiter. Das Byte für diesen Befehl errechnet sich aus der Anzahl plus >7F:
Z.B. für COL+ 21 bzw. . . . 21 < schreiben sie 94 (hex).

ROW+ Anzahl
... Anzahl
Die aktuelle Printposition wird um max. 32 Stellen nach unten weiter geschoben. Bei Überschreitung des unteren Bildrandes geht es oben eine Spalte rechts weiter. Das Byte für diesen Befehl errechnet sich aus der Anzahl plus > 9F:
Z.B. für ROW+ 21 bzw. ... 21^ schreiben Sie B4 (hex).

FOR Anzahl
... FOR Anzahl
Auf den Substack wird eine Schleife gesetzt. Näheres siehe END FOR. Es können fast beliebig viele Schleifen eröffnet werden, da mit dem Stack gearbeitet wird. Das Byte für diesen Befehl errechnet sich aus der Anzahl plus >BF:
Z.B. für FOR 3 bzw. ... FOR 3 schreiben Sie C2 (hex).

HSTR Anzahl, CPU-Adresse
... Anzahl ('§Adresse')
Ein String von max. 28 (!) Zeichen Länge wird horizontal auf den Bildschirm geprinted. Der String liegt dabei im CPU-RAM. Ein allgemeiner Adreßoperand, der immer als CPU-Adresse interpretiert wird, zeigt auf ihn. Das Byte für diesen Befehl errechnet sich aus der Anzahl plus >DF:
Z.B. für HSTR 5,§>8312 bzw. ... 5('§ > 8312') schreiben Sie E412 (hex).

FEND und END FOR GROM-Adresse
... END FMT und
... END FOR GROM-Adresse
Mit END FOR wird eine Schleife geschlossen, wobei die Adresse angibt, wohin der Interpreter springen soll, wenn die Schleife noch nicht beendet ist.

Beispiel:

```
G6187 FOR5
G6188 HTEX
      'HALLO'
G618E COL+ 27
G618F END FOR
      GROM§
      > 6188
```

Diese Routine schreibt das Wort "HALLO" 5-mal untereinander, ab einer durch YPT und XPT bestimmten Position. In G6187 wird dazu der Wert 5 auf den Substack gelegt und in G618F solange heruntergezählt, bis er Null und somit die Schleife beendet ist. Dabei wird dann die Schleife wieder vom Substack genommen. Es sind, wie gesagt, auch ineinander geschachtelte Schleifen möglich, z.B.:

```
G9723 FOR 3
G9724 FOR 7
G9725 HTEX
      'HALLO'
G972B COL+ 27
G972C END FOR
      GROM§
      > 9725
G972DCOL+3
G972E ROW+ 17
G972F END FOR
      GROM§
      > 9724
```

Übungsaufgabe: Finden Sie heraus, was diese Routine macht!

Das Byte für END FOR lautet > FB und ihm ist die 2 Byte lange GROM-Adresse nachgestellt. Das gleiche Byte hat aber auch FEND. Mit FEND verlassen Sie den Subinterpreter wieder und kehren in das normale GPL zurück. FEND ist nichts weiter als ein END FOR ohne die GROM-Adresse, bei dem kein Schleifenzähler mehr auf dem Substack liegt.

SCRO Offset
ROM-Listing: nicht vorhanden
Sämtliche auf den Bildschirm zu printende Zeichen werden vorher mit einem Screenoffset versehen, der in TI-Basic z.B. den Wert 96 und im E/A den Wert 0 hat. Mit SCRO verändern Sie den Offset, der beim Aufruf von FMT automatisch auf 0 gesetzt wird. Das Byte für diesen Befehl lautet > FC und ihm ist das Offset-Byte nachgestellt.

SCRO CPU-Adresse
... BIAS=(§Adresse)
Auch hier wird der Offset in gleicher Weise verändert. Das Byte für diesen Befehl lautet > FD und ihm ist ein allgemeiner Adreßoperand, der immer als CPU-Operand interpretiert wird, nachgestellt.

ROW Zeile
... YPT=Zeile
Der YPT wird auf einen neuen Wert gesetzt. Das Byte für diesen Befehl lautet > FE und ihm ist ein Byte von 0 bis 23 nachgestellt.

COL Spalte
... YPT=Spalte
Der YPT wird auf einen neuen Wert gesetzt. Das Byte für diesen Befehl lautet > FF und ihm ist ein Byte von 0 bis 31 nachgestellt.

Das Demoprogramm: "Einer gegen Waldi"
Diesem Reaktionstest liegt eine völlig neue Spielidee zugrunde und Ähnlichkeiten mit bereits bestehenden Spielen sind ganz bestimmt rein zufällig! Ihr Hund Waldi hat sich von seiner Leine losgerissen und läuft jetzt kreuz und quer über die Wiese. Sie wollen ihn mit Ihrer Spielfigur, gesteuert mittels Joystick 1, natürlich wieder einfangen. Das Spiel geht über 9 Runden und Sie spielen gegen die Uhr. Zu bemerken wäre dann nur noch, daß Ihre Figur mit doppelter Geschwindigkeit läuft, wenn Sie die Feuertaste drücken.
Aber nun zum Programm selbst. Wir beginnen mit allen Daten, die das Programm benötigt: Header, Chardaten für die Graphik, Spritewerte, dem Wert für das VDP-REG 1 und den Texten:

```
G6000 AA00 0000 0000
6010
G6008 0000 0000 0000
0000              Header

G6010 0000 608D 0557
414C
G6018 4449      "Waldi"

G601A 2020 0242 5252
1110  Char für die Wiese

G6022 0000 1070 8867
1008       Char für Waldi
G602A 0702 0202 0400
0000
G6032 0000 0000 00F1
0E10
G603A E020 2020 4000
0000

G6042 0106 0808 0601
0305  Char für den Mann
G604A 0901 0101 0204
0810
G6052 8060 1010 6080
C0A0
G605A 9080 8080 4020
1008

G6062 5050 800E 70B0
8406  Spritewerte Waldi
              und Mann

G606A C000 8800
       Spritewerte für die
       unbenutzten Sprites

G606E E2 Wert für VDP-
              REG 1

G606F 2A20 5045 4348
2047      TEXT '*PECH
             GEHABT *'
G6077 4548 4142 5420
2A

G607E 2A20 4752 4154
554C  TEXT '* GRATU-
            LATION*'
```

```
G6086 4154 494F 4E20
      2A
```

Nun müssen wir den Bildschirm aufbauen und die Werte für die Sprites setzen:

```
G608D 040C  BACK12
            Hintergrundfarbe
            dunkelgrün
G608F BE74  ST§>8374,1
            Tastaturmodus 1
      01
G6092 3100  MOVE
            > 0008
            TO VDP
            § > 0C00
      08AC  FROM
      0060  GROM§
      1A    > 601A
            Char für Wiese
            als CHR$ (128) in
            Charakterdefinitions-
            tabelle (beginnt bei
            V0800)
G6099 3100  MOVE
            > 0040
            TOVDP
            § > 0400
      40A4  FROM
            GROM
            § > 6022
      0060
      22
            Char für Sprites
            in Spritedefinitions-
            tabelle (ab V0000)
G60A0 BEA3  ST VDP
            § > 0390,
            > C3
      90C3
            Wiesenfarbe in Farb-
            tabelle (beginnt
            bei V0380)
G60A4 0780  ALL128
            Rasenfläche auf Bild-
            schirm
G60A6 BEA0  ST VDP
            § > 003D,
            57
      3D39
            Rundenzähler setzen
G60AA 3900  MOVE
            > 0001
            TO REG
            >01
      0101  FROM
      606E  GROM
            § > 606E
            VDP-REG 1:
            Magnify 3
G60B0 3100  MOVE
      08A3  > 0008
      0060  TO VDP
      62    §>0300
            FROM
            GROM
            § > 6062
            Beide Sprites setzen
```

Mit der folgenden Schleife bekommen alle restlichen Sprites den ASCII-Wert 136 zugewiesen. Dieses Zeichen ist hier eine Leerstelle und das ist wichtig, da die Sprites sonst die COINC-Abfrage stören würden, auch wenn sie sich außerhalb des Bildschirms befinden:

```
G60B7 BF00  DST
      0308  § > 8300,
            > 0308
G60BB 3100  MOVE
      04B0  > 0004
      0060  TOVDP
      6A    * > 8300
            FROM
            GROM
            § > 606A
G60C2 A201  ADD
      04    §>8301,4
G60C5 D601  CEQ
      80    § > 8301,
            > 80
G60C8 40BB  BR
            GROM
            § > 60BB
```

In > 837A steht die Nummer des höchstwertigen Sprites, das von der Interruptroutine bewegt werden soll:

```
G60CABE7A  ST § >
      02   837A,2
```

Zeitzähler setzen:

```
G60CDBEA 0 ST
      3A39 VDP §
           >003A,
           57
```

> 8300 und > 8301/02 enthalten zwei weitere Zähler: Immer wenn > 8301/02 von > 0080 auf > 0000 heruntergezählt wurden, bekommt Waldi neue Bewegungsdaten zufällig zugewiesen und > 8300 wird heruntergezählt. Hat > 8300 den Wert > 00 erreicht, so wird der Zeitzähler auf dem Bildschirm heruntergezählt. Hat dieser den Wert 48 erreicht (das ist der ASCII-Wert für die Null), so erfolgt die Ausgabe "PECH GEHABT" und eine neue Runde beginnt:

```
G60D1  BE00  ST §
       04    >8300,
             04
             Zähler 1
G60D4  02FF  RAND
             > FF
             Zufällige Bew. in
             Y-Richtung
G60D6  DE78  SRA§>
       01    8378,1
             RAND-Wert durch 2
G60D9  BCA7  ST
       8078  VDP §
             >0780,
             §>8378
G60DD  02FF  RAND
             > FF
             Zufällige Bew. in
             X-Richtung
G60DF  DE78  SRA §>
       01    8378,1
             RAND-Wert durch 2
G60E2  BCA7  ST
       8178  VDP
             §>0781,
             §>8378
G60E6  DA85  CLOG
       0220  §>8802,
             > 20
             COINC-Abfrage anhand
             des VDP-Statusbyte
             (siehe Handbuch zum
             Ed/Ass!)
G60EA  412D  BR
             GROM
             §>612D
G60EC  BF01  DST
       0080  §>8301,
             > 0080
             Zähler 2
```

Nun folgt die Tastaturabfrage. Der Joystick gibt die Werte +4, 0 und -4 in den Bytes > 8376 und > 8377 zurück. Sie werden mit 8 und bei gleichzeitigem Drücken der Feuertaste insgesamt mit 16 multipliziert. Der Y-Wert muß zusätzlich negiert (d.h. mit -1 multipliziert) werden, da der Mann sonst immer genau in die entgegengesetzte Richtung laufen würde:

```
G60F0  03    SCAN
             Tastatur- und
             Joystickabfrage
G60F1  E276  SLL §>
       03    8376,3
             Y-Wert mal 8
G60F4  E277  SLL §
       03    >8377,3
             X-Wert mal 8
G60F7  D675  CEQ §>
       12    8375,18
             Feuertaste ?
G60FA  4102  BR
             GROM
             §>6102
             Nein: Weiter
G60FC  E276  SLL §>
       01    8376,1
             Y-Wert mal 2
G60FF  E277  SLL§ >
       01    8377,1
             und X-Wert mal 2
G6102  8276  NEG
             §>8376
             Y-Wert mal -1
G6104  BDA7  DST
       8476  VDO §
             >8376
             Werte in die Sprite-
             motiontabelle schreiben
             (ab V0780)
G6108  DA85  CLOG
       0220  §>8802,
             >20
             COINC-Abfrage anhand
             des VDP-Statusbyte
G610C  412D  BR
             GROM
             §>612D
G610E  9301  DDEC
             §>8301
             Zähler 2 minus 1
G6110  40F0  BR
             GROM
             §>60F0
             Null?
G6112  9200  DEC
             §>8300
             Zähler 1 minus 1
G6114  40D4  BR
             GROM
             §>60D4
             Null?
G6116  0600  CALL
       34    GROM
             §>0034
             Ja: Beep und
G6119  92A0  DEC
       3A    VDP
```

```
            §>003A
Zeitzähler auf Bildschirm
             minus 1
G611C  D6A0 CEQ
       3A30 VDP>
            §003A,
            48
48 (=ASCII für 0)?
G6120  40D1 BR
            GROM
            §>60D1
Nein: Weiter spielen
G6122  0600 CALL
       36   GROM
            §>0036
Ja: Bad-Response-Tone
                 und
G6125  3100 MOVE
       0F03 >000F
       606F TO
            § 8303
            FROM
            GROM
            §>606F
Text "PECH GEHABT"
G612B  4136 BR
            GROM
            §>6136
              Weiter
G612D  0600 CALL
       34   GROM
            §>0034
Hund wurde gefangen
G6130  3100 MOVE
       0F03 >000F
       607E TO
            §>8303
            FROM
            GROM
            §>607E
Text "GRATULATION"
G6136  08   FMT
     Text darstellen
G6137  FE0A ROW10
            YPT=10
G6139  FF08 COL8
            XPT=8
G613B  4E2A HCHA
            15, "*"
"**************"
G613D  90   COL+
            17
     XPT=XPT+17
G613E  002A HTEX
            "*"
              "*"
G6140  4C20 HCHA
            13,"
               ""
G6142  002A HTEX
            "*"
              "*"
G6144  90   COL
```

```
              +16
       XPT=XPT+17
G6145  EE03 HSTR
            15,§
            >8303
  "* Textstring *"
G6147  90   COL+
            17
       XPT=XPT+17
G6148  002A HTEX
            "*"
              "*"
G614A  4C20 HCHA
            13, "
               ""
G614C  002A HTEX
            "*"
              "*"
G614E  90   COL
            +17
       XPT=XPT+17
G614F  4E2A HCHA
            15, "*"
"***************"
G6151  FB   FEND
G6152  8CA0 PUSH
       3A   VDP
            §>003A
Erreichte Punktzahl
         auf Stack
G6155  92A0 DEC
       3D   VDP>
            003D
Rundenzähler minus 1
G6158  BF0I DST
       0A00 §>8301,
            >0A00
      Warteschleife
G615C  9301 DDEC
            §>8301
G615E  415C BR
            GROM
            §>615C
G6160  08   FMT
        Text löschen
G6161  FE0A ROW10
            XPT=10
G1663  FF08 COL8
            XPT=8
G6165  C4   FOR5
Schleife: Das Folgende
                  5x
G6166  4E80 HCHA
            15,128
CALL HCHAR (YPT,
      XPT, 128,15)
G6168  90   COL+
            17
       XPT=XPT+17
G6169  FB61 END
       66   FOR
            GROM
            §>6166
```

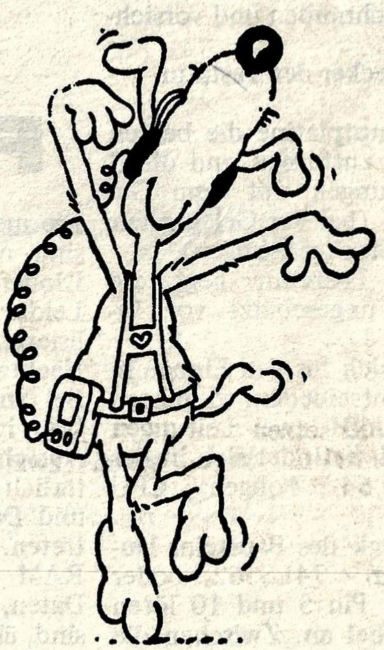
```
Weiter erst, wenn die
letzten beiden Zeilen 5x
 abgearbeitet wurden
G616C  FB   FEND
G616D  D6A0 CEQ
       3D30 VDP>
            §003D,
       4    8
Alle 9 Runden gespielt?
G6171  40CD BR
            GROM
            §>60CD
Nein: Weiter spielen
```

Zum Schluß des Spiels erscheint die Punktetabelle auf dem Bildschirm:

```
G6173  0720 ALL32
G6175  08   FMT
G6176  FE02 ROW2
G6178  FF09 COL9
G617A  0C2A HTEX
       2047 "*
            GAME
            OVER
            *"
       414D 4520
            4F56
       4552 202A
G6188  FE05 ROW5
G618A  FF02 COL2
G618C  1049 HTEX
            'Ihre
            Punkt-
            zahlen:'
```

```
       4852
       4520 5055
            4E4B
       545A 414B
            4C45
       4E3A
G619E  FE15 ROW21
G61A0  FF02 COL2
G61A2  1146 HTEX
            'FCTN
            + für
            Ende!'
       4354
       4E20 2B20
            2046
       5545 5220
            454E
       445  21
G61B5  FB   FEND
G61B6  BF00 DST
       00FA §>8300,
            >00FA
G61BA  BDB0 ST
       0090 VDP
       7C   *>8300,
            *>837C
G61BF  A601 SUB §
       03   8301,3
G61C2  D601 CEQ
       DF   §>8301,
            >DF
G61C5  41BA BR
            GROM
            §>61BA
G61C7  41C7 BR
            GROM
            §>61C7
```

Sven Dyroff



RESET-TASTER FÜR DEN TI 99/4A

Viele Programmierer schaffen es immer wieder, den TI in einen Mode zu schicken, welcher nur durch Aus- und Einschalten der Systemkonsole wieder verlassen werden kann.

Dieses lästige Aus- und Einschalten kann durch einen kleinen Systemeingriff vermieden werden, zumal dies sowohl den Netzschalter als auch die Einzelkomponenten schont.

Die Lösung dieses Problems heißt Reset-Taster.

Zwei Möglichkeiten zum Einbau stehen an:

1. Direkter Einbau in die Konsole (Loch bohren usw.)
2. Anschluß an den freien Pins vom Joystickport

KONSOLE ODER JOYSTICK– BEIDES IST MÖGLICH

Bei beiden Möglichkeiten ist wie folgt vorzugehen:

- Konsole auf den Rücken legen
- Die 7 Schrauben lösen und entfernen
- Unterboden abheben (auf Ein-/Ausschalter aufpassen!)
- Die 2 Netzteilschrauben lösen und die Spannungsversorgungsplatine entfernen
- Hauptplatine von den 3 Befestigungen losschrauben und vorsichtig anheben.
- Anschlußstecker der Tastatur entfernen.
- An der Hauptplatine die beiden Metallbügel entfernen und die 3 Verschraubungen mit dem Gehäuse lösen (bei der Gelegenheit Gromport-Stecker reinigen).

Nach dieser Prozedur liegt die Hauptplatine ungeschützt vor Ihnen.

Haben Sie sich für den Einbau in die Konsole entschieden, so müssen Sie nun an U601 zwei Leitungen anlöten (U601 befindet sich in der Nähe der 64 Poligen CPU TMS9900).

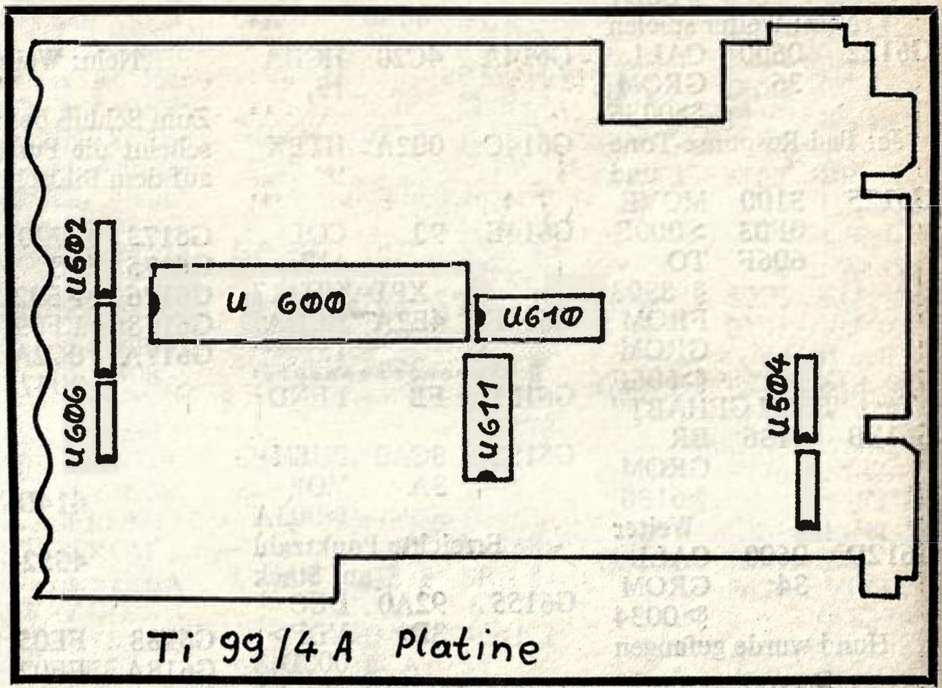
Der Aufdruck des Bausteins lautet entweder 74LS362 oder TIM9904. An Pin 5 und 10 löten Sie je ein Kabel an. Zwischen die Kabel einen Taster und fertig ist der Reset Taster.

Wer sich für den Joystickport entschieden hat, geht wie folgt vor.

- Kabel an Pin 5 U601 löten
- Am Joystickport die beiden freien Pins suchen
- Einen Pin auf Masse legen (ist unmittelbar daneben)
- Kabel von Pin 5 an den zweiten freien Pin vom Joy-Port löten

Jetzt muß nur von außen eine 9-polige Buchse mit Taster angesteckt werden und die Sache ist erledigt.

Den Zusammenbau der Konsole sollten Sie natürlich nicht vergessen.



BATTERIE-PUFFERUNG FÜR CMOS-RAM

Die meisten Schaltungen dieser Art sind mit einer Batterie und zwei Dioden aufgebaut.

Leider beinhaltet diese Art der Realisierung aber einen entscheidenden Nachteil.

Bei einem plötzlichen Spannungsabfall im Computersystem (durch Ausschalten der Power) können natürlich auf allen Steuer-, Adress- und Datenleitungen Störungen auftreten. Dies hat zur Folge, daß im RAM einzelne Speicherstellen mit Daten, die zufällig auf dem Bus sind, überschrieben werden.

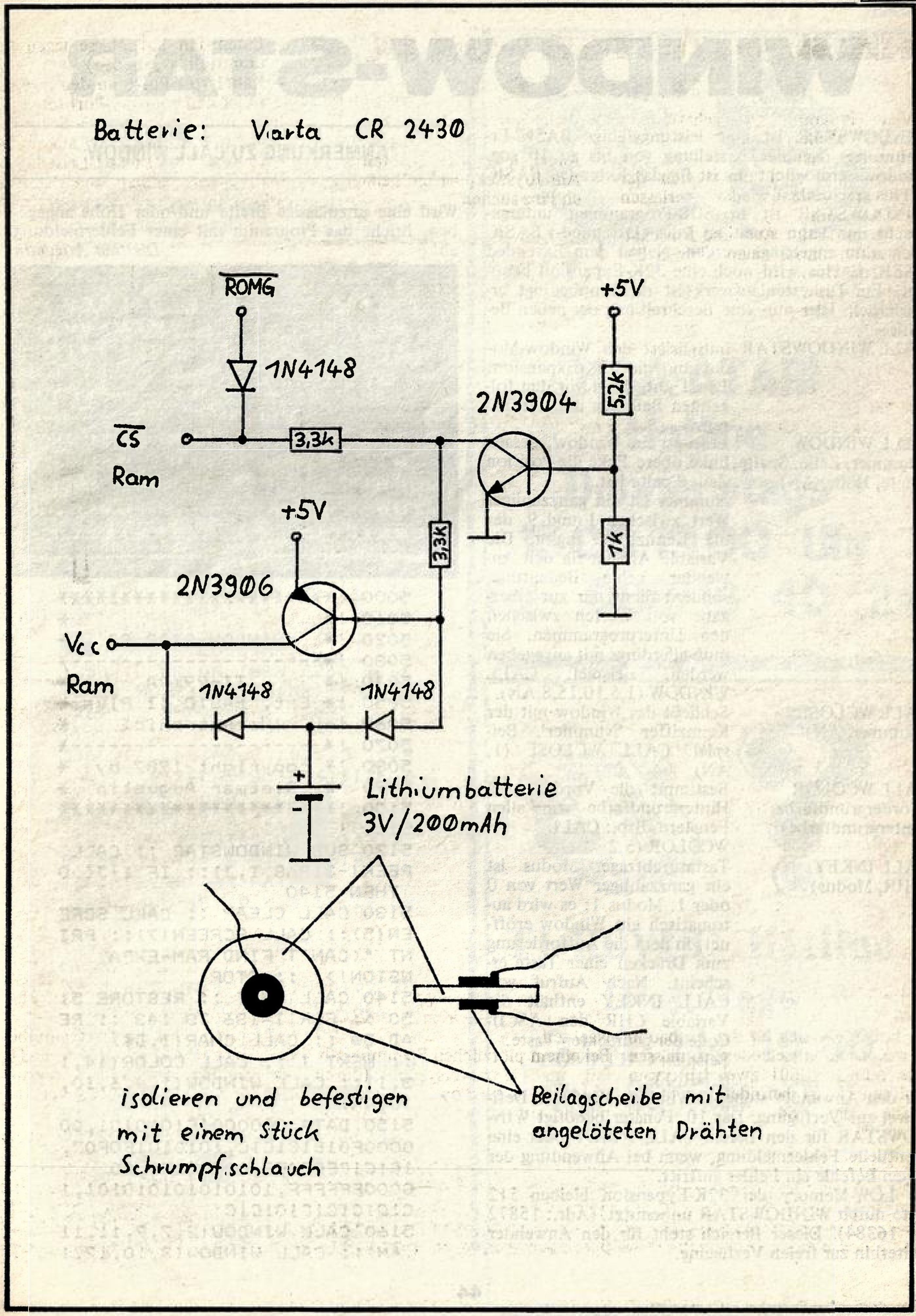
Die vorliegende Hardware beseitigt diesen Makel.

Sie besteht im Wesentlichen aus schnellen Schalttransistoren, die bei Power Down die Chip Select Leitung des RAM-Bausteins auf High (> 2.4 Volt) zieht.

Dadurch wird ein Überschreiben der Speicherzellen verhindert. Die Verwendung des ROMG Signals läßt auf den Einsatz im Bereich Hex 6000 schließen.

Natürlich können auch die Chip Select Signale anderer Anwendungsmöglichkeiten (GRAM Modul) anstatt des Signals ROMG (Aktiv Low) verwendet werden.

Reiner Ziegler


Batterie: Varta CR 2430
ROMG
1N4148
+5V
2N3904
5,2k
CS
Ram
3,3k
+5V
3,3k
1k
2N3906
Vcc
Ram
1N4148
1N4148
Lithiumbatterie
3V/200mAh
isolieren und befestigen
mit einem Stück
Schrumpfschlauch
Beilagscheibe mit
angelöteten Drähten

WINDOW-STAR

WINDOWSTAR ist eine leistungsfähige BASIC-Erweiterung, das die Darstellung von bis zu 10 sog. Windows ermöglicht. Es ist für das Extended BASIC II Plus geschrieben.
WINDOWSTAR ist in SUB-Programmen untergebracht und kann somit an jedes (Extended-) BASIC Programm angehängt werden. Neben dem Extended BASIC II Plus wird noch eine 32K-Expansion benötigt. Ein Diskettenlaufwerk ist nicht unbedingt erforderlich. Hier nun eine Beschreibung der neuen Befehle:

CALL WINDOWSTAR — initialisiert den Window-Modus und die 32K-Expansion. Es ist unbedingt vor den folgenden Befehlen durchzuführen.

CALL WINDOW (Nummer, Zeile, Spalte, Breite, Höhe, AN) — Erzeugt ein Window, dessen linke obere Ecke die Position Zeile/Spalte hat.
Nummer ist ein ganzzahliger Wert zwischen 1 und 9, der die Kennziffer angibt. Die Variable AN hat für den Anwender keine Bedeutung, sondern dient nur zur Übergabe von Werten zwischen den Unterprogrammen. Sie muß allerdings mit angegeben werden. Beispiel: CALL WINDOW (1,5,10,15,8,AN)

CALL WCLOSE (Nummer, AN) — Schließt das Window mit der Kennziffer ‘Nummer’. Beispiel: CALL WCLOSE (1, AN)

CALLWCOLOR (Vordergrundfarbe, Hintergrundfarbe) — Bestimmt die Vorder- und Hintergrundfarbe von **allen** Fenstern. Bsp.: CALL WCOLOR (5,2

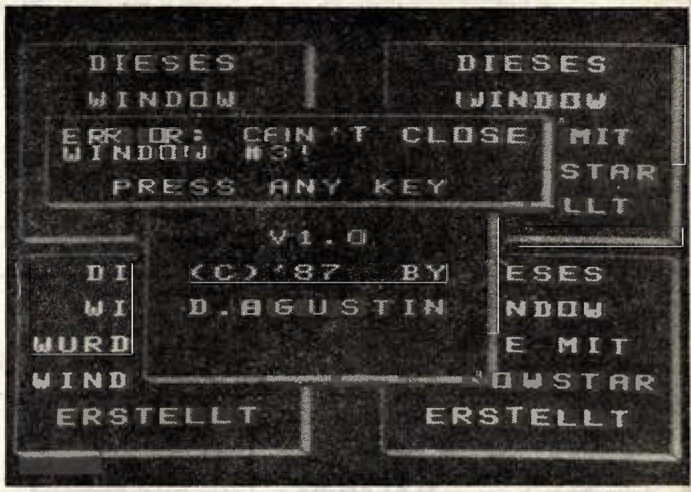
CALLINKEY (CHR, Modus) — Tastaturabfrage. Modus ist ein ganzzahliger Wert von 0 oder 1. Modus 1: es wird automatisch ein Window eröffnet, in dem die Aufforderung zum Drücken einer Taste erscheint. Nach Aufruf von CALL INKEY enthält die Variable CHR den ASCII-Code der gedrückten Taste.
Beispiel: CALL INKEY (CHR, 1)

Für den Anwender stehen 9 Windows zur freien Definition zur Verfügung. Das 10. Fenster benötigt WINDOWSTAR für den Befehl CALL INKEY oder eine eventuelle Fehlermeldung, wenn bei Anwendung der neuen Befehle ein Fehler auftritt.
Im LOW-Memory der 32K-Eypansion bleiben 512 Byte durch WINDOWSTAR unbenutzt. (Adr.: 15872 bis 16384). Dieser Bereich steht für den Anwender weiterhin zur freien Verfügung.

ANMERKUNG ZU CALL WINDOW:

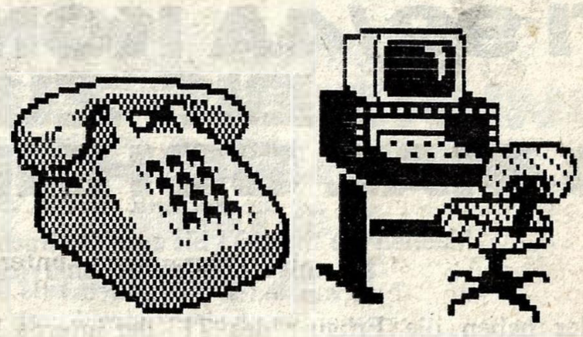
Wird eine unzulässige Breite und/oder Höhe angegeben, bricht das Programm mit einer Fehlermeldung ab!

Dietmar Augustin

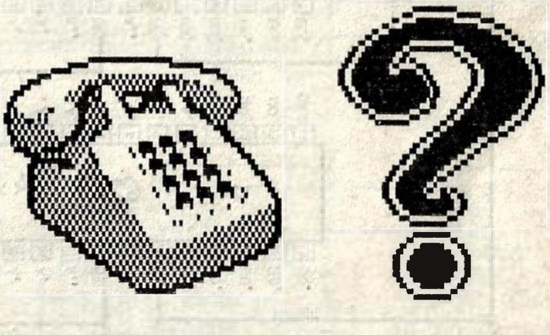



```
5000 !**********************
5010 !*                    *
5020 !*   WINDOW-STAR 99   *
5030 ! * -----------------*
5040 !*      TI-99/4A      *
5050 !* Ext. BASIC II Plus *
5060 !*    (Mechatronic)   *
5070 ! * -----------------*
5080 !* copyright 1987 by  *
5090 !*  Dietmar Augustin  *
5100 !**********************
5110 !
5120 SUB WINDOWSTAR :: CALL
PEEK(-31868,I,J):: IF I+J<>0
 THEN 5140
5130 CALL CLEAR :: CALL SCRE
EN(5):: CALL SCREEN(7):: PRI
NT "<CAN'T FIND RAM-EXPA
NSION!>" :: STOP
5140 CALL INIT :: RESTORE 51
50 :: FOR I=136 TO 143 :: RE
AD D$ :: CALL CHAR(I,D$)
:: NEXT I :: CALL COLOR(14,1
3 , 1):: CALL WINDOW(1,4,6,10,
10,AN)
5150 DATA 0000001F1010101,00
0000F0181C1C1C,1010101F0F07,
1C1C1CECF4F8,000000FF,00
0000FFFFFF,1010101010101010,1
C1C1C1C1C1C1C1C
5160 CALL WINDOW(2,7,9,11,11
,AN):: CALL WINDOW(3,10,12,1
```

```
2,12,AN):: DISPLAY AT(13
,12)SIZE(10):"WINDOWSTAR" ::
 DISPLAY AT(15,16)SIZE(4):"V
1.0"
5170 DISPLAY AT(17,12)SIZE(1
0):"(C)'87  BY" :: DISPLAY A
T(19,12)SIZE(10):"D.AUGU
STIN" :: CALL WAIT(250):: CA
LL WCLOSE(3,AN):: CALL WCLOS
E(2,AN)
5180 CALL WCLOSE( 1,AN):: SUB
END
5190 SUB WINDOW(NR,ZE,SP,ZL,
SL,AN):: ADR=768*NR+7424
5200 CALL MOVE(2,0,ADR,768 ):
: CALL HCHAR(ZE,SP,136):: CA
LL HCHAR(ZE,SP+ZL+1,137)
5210 CALL HCHAR(ZE+SL+1,SP,1
38):: CALL HCHAR (ZE+SL+1,SP+
ZL+1,139):: CALL HCHAR(Z
E,SP+1,140,ZL):: CALL HCHAR(
ZE+SL+1,SP+1,141,ZL)
5220 CALL VCHAR(ZE+1,SP,142,
SL):: CALL VCHAR(ZE+1,SP+ZL+
1,143,SL):: FOR I=ZE+1 T
O ZE+SL :: CALL HCHAR(I,SP+1
,32,ZL):: NEXT I
5230 AN=AN+1 :: SUBEND
5240 SUB WCLOSE (NR,AN)
5250 IF NR<AN THEN 5260 ELSE
 CALL MOVE(3,(768*NR)+7424,0
,768):: AN=AN-1 :: SUBEX
IT
5260 CALL WINDOW(10,5,5,18,4
,AN):: DISPLAY AT(6,4)SIZE(1
8):"ERROR: CAN'T CLOSE"
5270 DISPLAY AT(7,4)SIZE(18)
:"WINDOW #"&STR$(NR)&"!" ::
DISPLAY AT(9,6)SIZE(13)B
EEP:"PRESS ANY KEY"
5280 CALL KEY(0,K,S):: IF S=
0 THEN 5280
5290 NR=10 :: GOTO 5250 :: S
UBEND
5300 SUB INKEY(CHR,MOD):: IF
 MOD=0 THEN 5320 ELSE CALL W
INDOW(10,10,10,16,2,AN)
5310 DISPLAY AT(11,9)SIZE(16
):"PLEASE PRESS ANY" :: DISP
LAY AT(12,11)SIZE(13)BEE
P:"KEY TO GO ON!"
5320 CALL KEY(0,CHR,S):: IF
S=0 THEN 5320
5330 CALL WCLOSE(10,AN):: SU
BEND
5340 SUB WCOLOR(FC,BC):: CAL
L COLOR(1,FC,BC,14,FC,BC)::
SUBEND
```

HCA-ATARI
Hotline
Donnerstags
15.00-19.00 Uhr
Tel.: 089/18 40 24

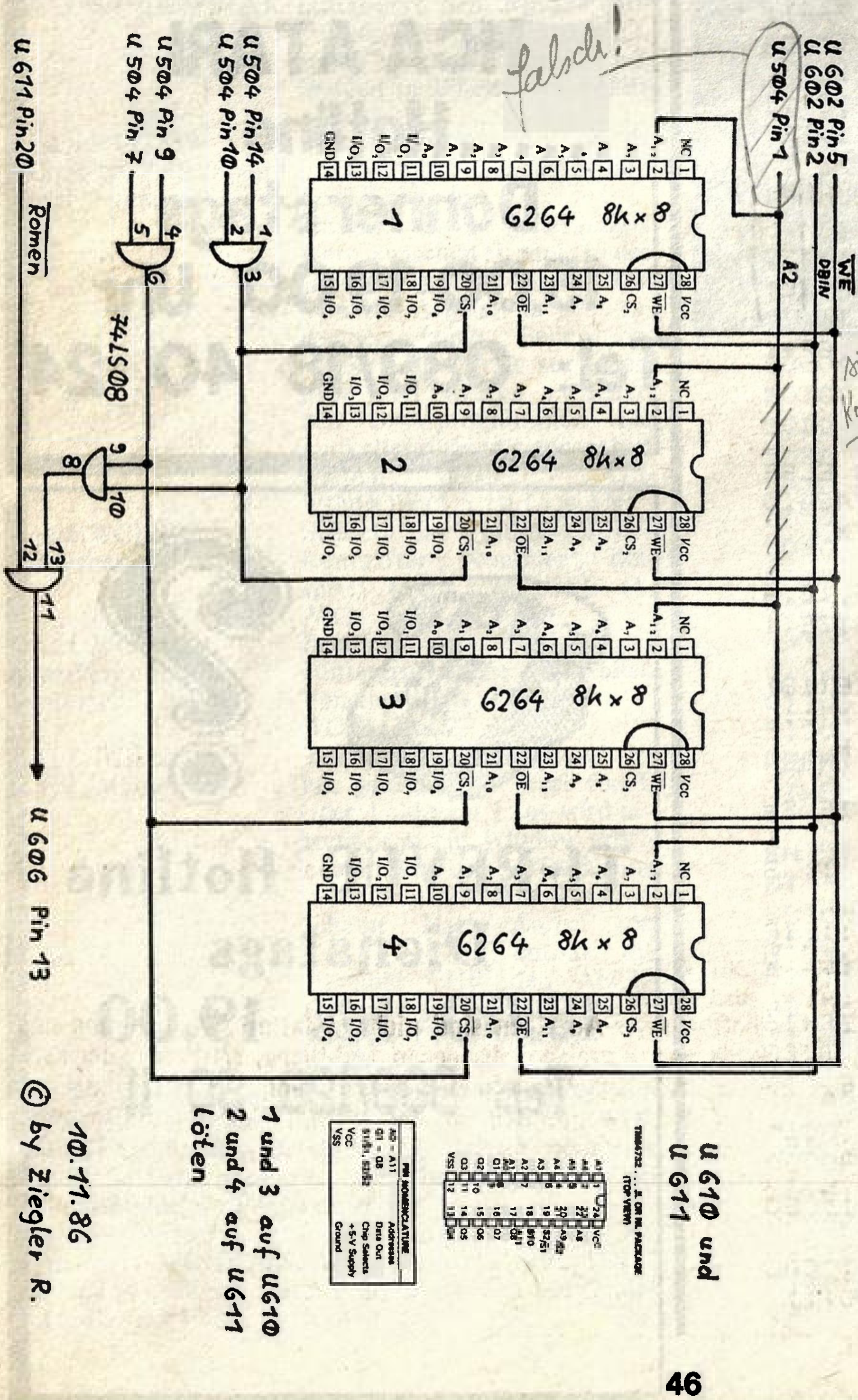


TI-REVUE Hotline
Dienstags
15.00 bis 19.00
Tel.: 089/129 80 11

TI 99/4A KONSOLE BIS ZU 3 MAL SCHNELLER

Machen Sie Ihren TI bis zum Dreifachen schneller, mit einem einfachen Trick!

Leider haben die Erbauer des TI den schnellen 16Bit Datenbus auf 8Bit reduziert und somit die enormen Geschwindigkeitsvorteile verschenkt. Glücklicherweise wurde der interne Bus auf 16Bit Breite belassen. Greift der TI auf externe Erweiterungen (32K RAM, DSR, GROM PORT usw.) zu, so sorgt eine ausgeklügelte Elektronik für die Einfügung der entsprechenden Wartezyklen.



Die Reduzierung des Buses hat eine Verlangsamung des Systems um den Faktor drei im RAM oder ROM Bereich zur Folge.

Im Nachfolgenden möchte ich eine 32KByte Speichererweiterung beschreiben, die in die Konsole eingebaut wird und somit den vollen 16Bit Bus verwendet. Der Geschwindigkeitsvorteil im reinen Memory Bereich beträgt drei. Leider kommt dieser Vorteil aber nicht voll zum Tragen, da auch Bildschirmausgaben und GROM Zugriffe ausgeführt werden.

Die Geschwindigkeitssteigerung dürfte realistisch gesehen also im Bereich zwei bis drei liegen.

Im nachfolgenden Schaltplan wird die notwendige Verdrahtung gezeigt. Jeweils zwei Rambausteine werden übereinander gesteckt und Parallel verlötet (außer PIN 20 Chip select). Anschließend werden die beiden Ramblöcke auf die ROMS U610 und U611 gelötet und zwar so, daß die RAM Pins 1,2,28, 27 vorne überstehen. Leider sind die ROMS U610 und U611 nicht ganz Pinkompatibel zu den RAM Bausteinen, so daß die Pins 20,22 und 23 des jeweils untersten RAM Bausteins gesondert verbunden werden müssen. Die an RAM Pin23 notwendige Adressleitung A11 kann mit ROM Pin18 verbunden werden.

Vor dem Verdrahten der restlichen Logik sollte der PIN13 von U606 durchtrennt werden (kleiner Schnitt auf der Leiterbahn genügt). An diesem IC wird unser Signal eingespeist, welches ein Umschalten des 16Bit Buses auf einen 8Bit Bus verhindert.

ALTE SPEICHER-ERWEITERUNG MUSSWEG

Noch ein Tip am Rande. U504 trägt den Aufdruck 74LS138 und befindet sich auf der Leiterplatte ganz rechts. Dieser Hinweis nur, da links daneben noch ein 74LS138 liegt (siehe Bestückungsskizze).

Nach erfolgtem Umbau können Sie ihre 32KByte Speichererweiterung in der Peripheriebox entfernen und verkaufen. Die Verwendung beider Speicher geht nicht.

Reiner Ziegler

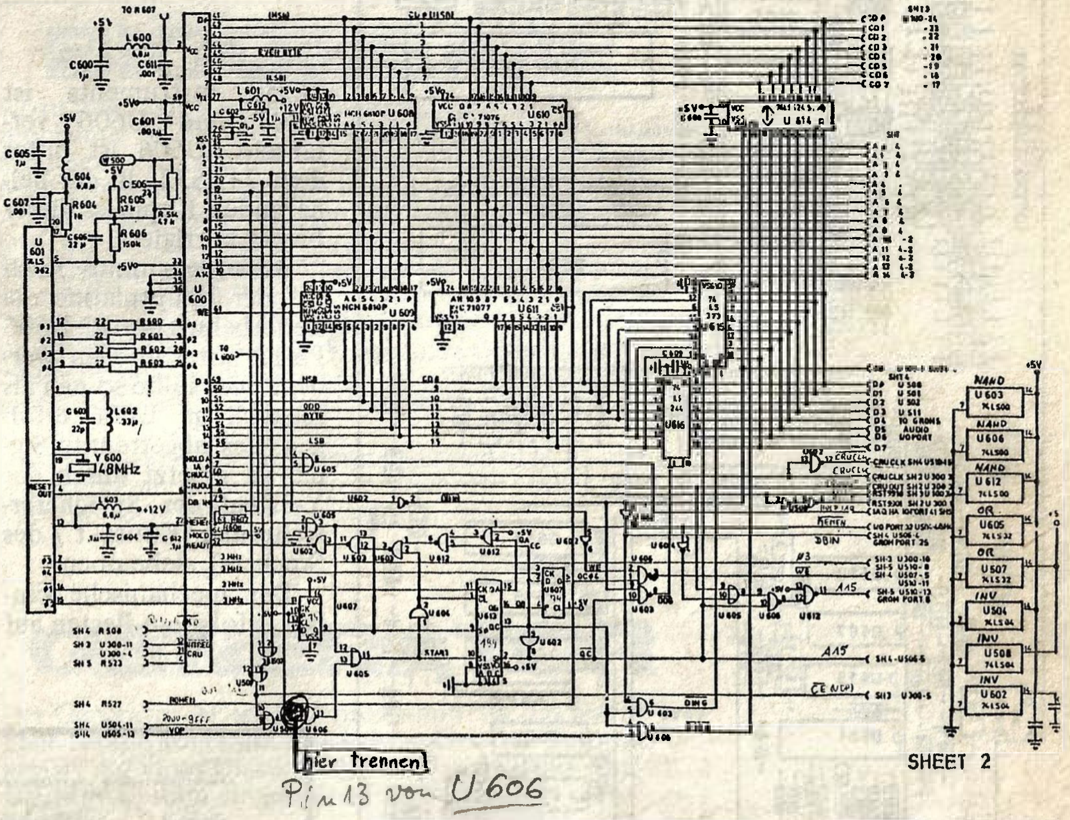
32K-BYTE CMOS SPEICHERERWEITERUNG MIT 16 BIT DATENBUS FÜR TI 99/4A VERSION 2.1

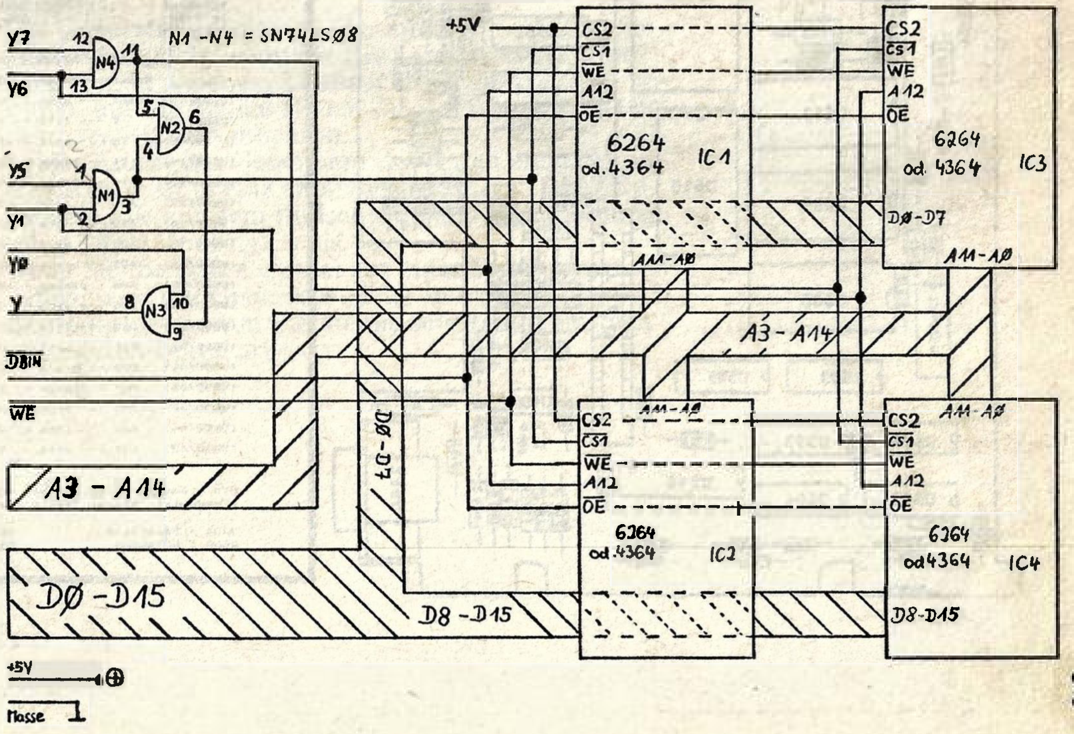
Diese Speichererweiterung unterscheidet sich in der Hauptsache von den bisher bekannten für den TI99/4A durch die 16 K x 16 Bit Speicherorganisation. Hierdurch kann die Geschwindigkeit in diesem RAM - Bereich verdoppelt werden, da der Wartetakt für die CPU entfä llt, der nötig ist, wenn der Prozessor bei der 8 Bit - Speicherversion zweimal auf den Speicher zugreifen muß, um ein Wort mit 16 Bit zu lesen oder zu schreiben. Softwaremäßig ist die Erweiterung kompatibel zu den anderen Versionen.

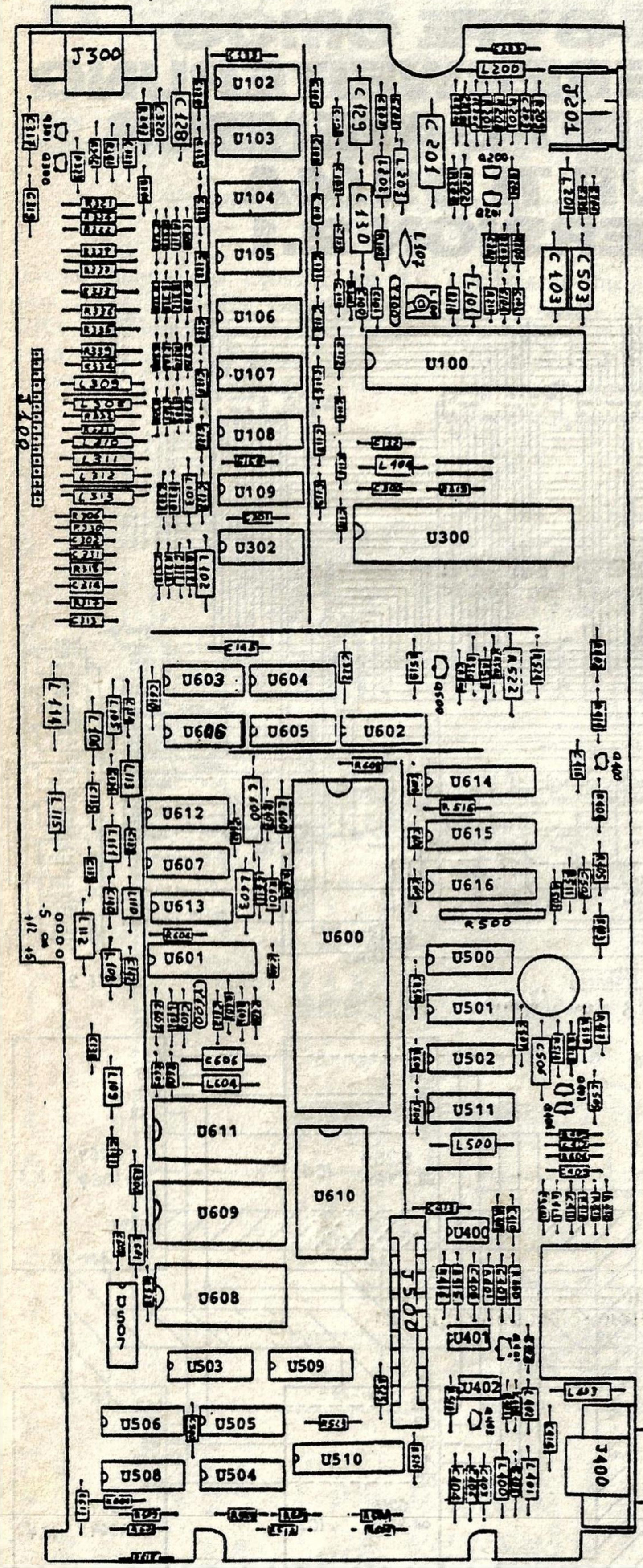
Der Anschluß kann nur in der Konsole erfolgen, da nur hier alle 16 Datenleitungen vorhanden sind. Zur Verbindung dient ein 40-poliges Flachbandkabel, das auf der Memoryplatine mit einem Stecker angeschlossen wird und dessen anderes Ende im Rechner direkt an den Beinen der ICs vorsichtig angelötet wird. Hierzu wird ein netzgetrennter Lötkolben mit einer sehr feinen Spitze benötigt.

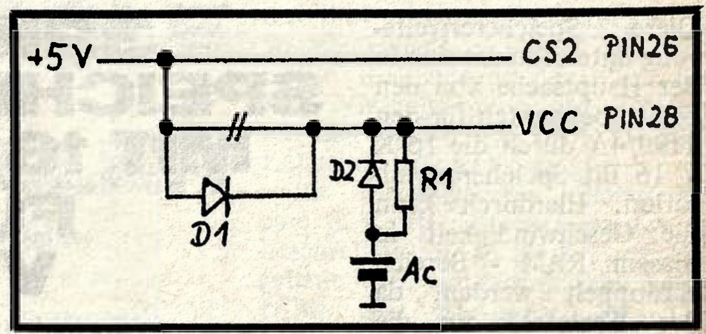
Die Datenleitungen D0 bis D15 sowie die Adressleitungen A3 bis A14 werden am einfachsten am TMS9900 (U600) selbst angeschlossen, die Spannungsversorgung (+5V, Masse) läßt sich am besten dort anlöten, wo das Kabel vom Netzteil her angeschlossen ist. Die Signale Y0, Y1, Y5, Y6 und Y7 werden am IC 74LS138 (U504) an den bei der Steckerbelegung angegebenen Pins abgegriffen. Ebenso sind die Leitungen mit den Signalen DBIN*, WE*, und Y an den im Bild genannten ICs und Pins vorsichtig anzulöten.

ACHTUNG: Im "Components Lay-Out" von









Texas Instruments ist U603 und U606 vertauscht. U606 ist unter dem 74LS32. In der beiliegenden Kopie ist dieser Fehler korrigiert.

Außerdem muß noch auf der Hauptplatine die Leiterbahn zum IC U606 Pin13 unterbrochen werden. Dies sollte so nah als möglich an U606 erfolgen, das abgetrennte Signal ist ersetzt durch das Y-Signal der Speichererweiterung (Kontakt 7 des Steckers), siehe oben.

Der mechanische Einbau erfolgt am Besten auf dem oberen Aluminiumdeckel der Hauptplatine. Die 32K-Erweiterung besteht aus:

- 1 doppelseitige Platine, alle ICs gesockelt
- 1 Digital-IC 74LS08
- 4 CMOS-Rams 8K*8 (HM 6264 LP15 od. uPD 4364 C15)
- 1 Tantalkondensator 6,3 uF/35V
- 1 keramischer Kondensator 100nF
- 1 40-poliger Pfostenverbinder
- 1 40-polige Kupplung
- 1 40-poliges Flachbandkabel

Anschlußbelegung Verbindungskabel 32K Platine <=> TI99/4A

Anschluß im TI99 (IC/ PIN)			Stecker 32K-Platine 1	2			Anschluß im TI99 (IC/ PIN)
	+5V	-- braun -	0==0		- rot -----	+5V	
	Masse	- orange -	0==0		- gelb ----	Masse	
(U508/10)	DBIN*	--- grün -	0	0	- blau ----	WE*	(U508/ 3)
(U606/13)	Y	--- lila -	0	0	- grau ----	Y7	(U504/ 7)
(U504/ 9)	Y6	--- weiß -	0	0	- schwarz -	Y5	(U504/10)
(U504/14)	Y1	-- braun -	0	0	- rot -----	Y0	(U504/15)
(U600/10)	A14	- orange -	0	0	- gelb ----	A13	(U600/11)
(U600/12)	A12	--- grün -	0	0	- blau ----	A11	(U600/13)
(U600/14)	A10	--- lila -	0	0	- grau ----	A09	(U600/15)
(U600/16)	A08	--- weiß -	0	0	- schwarz -	A07	(U600/17)
(U600/18)	A06	-- braun -	0	0	- rot -----	A05	(U600/19)
(U600/20)	A04	- orange -	0	0	- gelb ----	A03	(U600/21)
(U600/56)	D15	--- grün -	0	0	- blau ----	D14	(U600/55)
(U600/54)	D13	--- lila -	0	0	- grau ----	D12	(U600/53)
(U600/52)	D11	--- weiß -	0	0	- schwarz -	D10	(U600/51)
(U600/50)	D09	-- braun -	0	0	- rot -----	D08	(U600/49)
(U600/48)	D07	- orange -	0	0	- gelb ----	D06	(U600/47)
(U600/46)	D05	--- grün -	0	0	- blau ----	D04	(U600/45)
(U600/44)	D03	--- lila -	0	0	- grau ----	D02	(U600/43)
(U600/42)	D01	--- weiß -	0	0	- schwarz -	D00	(U600/41)
			39	40			

Pin 1 des Steckers ist mit einem Dreieck markiert.
* bedeutet: Signal "aktiv low".

U504 = SN74LS138 U600 = TMS9900
U508 = SN74LS04 U606 = SN74LS00

BIOCHEMIE

Wozu dient das Programm überhaupt?
Biochemie ist ein wichtiges Hilfsmittel für alle Schüler, Studenten etc., die mit biochemischen Strukturformeln arbeiten und die sich häufig durch den Berg von Abkürzungen ringen müssen, die es in der Biochemie gibt, z.B. ATP –Adenosintriphosphat.
Das Programm hält Strukturformeln von den wichtigsten Aminosäuren, Peptiden, Nucleotiden, Nucleinbasen, Nucleosiden, Monosacchariden, Di-Sacchariden und Fetten für Sie bereit. Weiter sind die wichtigsten Abkürzungen der Biochemie von A bis Z vorhanden und die Bauweise von Proteinen ist in einem ausgiebigen Text niedergeschrieben.
Eine Erklärung des Programms erübrigt sich, da alle Unterprogramme und Programmschritte erklärt sind. Diese werden vom S.S. unterstützt.

GENERATOR

Der Generator ist ein Hilfsprogramm für das eigentliche Hauptprogramm Biochemie. Der Generator generiert auf einer Diskette Daten, welche dann vom Hauptprogramm eingelesen werden, wenn es diese benötigt.
Der Generator arbeitet vollkommen selbständig, Sie müssen lediglich, wenn Sie dazu aufgefordert werden, eingeben, in welchem Laufwerk sich die Diskette befindet, auf die Sie die Daten, die der Generator generiert, abspeichern möchten.
Leider dauert das Generieren der Daten einige Minuten, aber es ist ja eine einmalige Prozedur. Die Zählvariable, die auf dem Bildschirm beim Generieren zu sehen ist, läuft übrigens bis 66.
Wenn Sie die Daten einmal auf einer Diskette generiert haben, wird der Generator nicht mehr benötigt, da das Hauptprogramm Biochemie selbständig mit den generierten Daten arbeitet.

```
100 !**********************
    *                    *
    *    BIOCHEMIE       *
    *    GENERATOR       *
    *                    *
110 !*        VERSION 1.0*
    *                    *
    *(C) 1986 BY DIRK    *
    *        JUNGHANS    *
    *        (DJIS)      *
120 !*                   *
    *SYSTEM:TI99/4A      *
    *EXTENDED BASIC      *
    *32KRAM              *
    *DSK1 OPT. DSK2      *
130 !*SPEECH SYNTHESIZER *
    *                    *
    *dirk junghans       *
    *am fort-biehler 9   *
    *6503 mz-kastel      *
140 !**********************
141 !
142 !VOR DEM LADEN "CALL FIL
ES(1)" UND "NEW" EINGEBEN
143 !
150 ON WARNING NEXT :: DIM T
$(24)
160 CALL SCREEN(3):: FOR I=0
 TO 9 :: CALL COLOR(I,5,15):
: NEXT I
170 DISPLAY AT(10,9)ERASE AL
L:"BIOCHEMIE" :: DISPLAY AT(
11,9):"GENERATOR"
180 DISPLAY AT(15,1):"    (C)
 1986 BY DIRK JUNGHANS
        (DJIS)"
190 DISPLAY AT(2,18):"PROGRA
MMED" :: DISPLAY AT(3,18):"B
Y DJIS"
200 CALL SAY("#PRESS##A1##KE
Y#")
210 CALL KEY(0,K,S):: IF S=0
 THEN 210 ELSE CALL CLEAR
220 GOSUB 1740 :: CALL CLEAR
230 DISPLAY AT(1,1)ERASE ALL
:" DIRK JUNGHANS  BIOCHEMIE
 "&RPT$("*",28)
240 B$=RPT$(" ",28)
250 !AMINOSAEUREN (ALLGEMEIN
),D-AMINOSAEURE&L-AMINOSAEUR
E
260 DATA ''''''''''''''''''''
'''''''''',''''''''''''''''''
'''''''''',''''''''''''''''''
''''''''''',D-AMINOSAEURE  L
-AMINOSAEURE
270 DATA (ALLGEMEIN)    (ALL
GEMEIN'''",''''''''''''''''
'''''''''',''''''''''''''''''
''''''''''',''''''''''''''''''
'''''''''''''
280 DATA o   oh         o  o
h'' ""'' ,'z y          z y
''' '"'' ',''c            c
'' ''"'" ,'' ' w
w'''''''''''''
290 DATA hxcxnh        hnxcxh
'''''''''' ,''w  a      a w'
'''' ''' ''' ,''r          r
''''''''''''',END
300 !GLYKOKOLL&L-ALANIN
310 DATA 'AMINOSAEUREN MIT E
INER''''''','REINEN C. H.-SEI
TENKETTE:'"',''''''''''''''''
''''''''''''''',GLYKOKOLL:
 L-ALANIN:'''
320 DATA ''''''''''''''''''''''
'''''''''',''''"'''''''''''''
''''''''''',''''''"'''''''''''
''''''''''''',''coo}
co o'} "'' ' '
330 DATA ''{ w            { w
''''''''',"hnxch{        hnx
cxh''''''',''b   a          b
 w'''''''' ' '''';''' ''' '''
''  c''h''''
340 DATA ''''''''''''''''''''''
'b,END
350 !L-VALIN&L-LEUCIN
360 DATA 'AMINOSAEUREN MIT E
INER''''','REINEN C. H.-SEIT
ENKETTE:'',"'''''''"'''''''''
''''''''''",L-VALIN:
L-LEUCIN:''',STOP
370 DATA '''ch                c
h''''''''''',''y z
w a''''''",hc   ch
 ch''''''''''','b     b
 y z '"''''''
380 DATA '''''''''''''''''''hc
 c h' ' ' ''''''',''''''"'' b '
'''b''''',END
390 !L-SERIN&L-THREONIN
400 DATA 'AMIN. MIT NICHT-IO
NISIERTEN,'ABER POLAREN GRUP
PEN IN DER,'SEITENKETTE:'''
''''''''''''',''''''''''''''''''
''''''''''''''
410 DATA L-SERIN:''''''''L-TH
REONIN:'',STOP
420 DATA '''chxoh          hxc
x o h''''''''',"' a
w'''''''''''',''''''''''''''''''''
'c  'H''''' ,"''''''''''''''''''
''''b''''''',END
430 !L-CYSTEIN&L-METHIONIN
440 DATA 'AMIN. MIT NICHT-IO
NISIERTEN','ABER POLAREN GRU
PPEN IN DER,'SEITENKETTE:'''
''''''''''''''',''''''''''''''''''
```

LISTINGS

```
''''''''''''''
450 DATA L-CYSTEIN:''''L-MET
HIONIN:'',STOP
460 DATA '''chxsh''''''''''c
h''''''''',''''''a
w a''''',''' ''''''''''h c x s
xc h''' '' ''' ,''''''''' b
a''''',END
470 !L-ASPARAGIN-&L-GLUTAMIN
-SAEURE
480 DATA 'SAURE AMINOSAEUREN
: ''''''',''''' '''''''' ''''
''''''''''',''' '''''''' '''''''
''''''''',L-ASPARAGIN- L
-GLUTAMIN-''
490 DATA ''SAEURE: SA
EURE:'''',STOP
500 DATA '''ch c
h''''''',''''w a
w a''''''',''''coo)
 c h'''''''',''''''''''''''''''
''w a'''''''
510 DATA '''''''''''''''''''''c
oo),END
520 !L-LYSIN&L-ARGININ
530 DATA 'BASISCHE AMINOSAEU
R E N:'' ''''''',''' '''' ''' ''
''''''''''',''''''''''''''''''
'''''''''''',L-LYSIN: L-
ARGININ:'''',STOP
540 DATA '''ch c
h''''''',''''w a
w a''''''',''''ch
 ch nh'''',''''w a
 w a w a'''
550 DATA '''ch hc
 c''''',''''w a(
az y l('''',''''chxnh
 n nh'',''''''a b
 h a',END
560 !PEPTIDBINDUNG
570 DATA 'PEPTIDE:''''''''''
'''''' ''',''''''' ''''''' '''
'''''''''''BAUPRINZIP DER P
EPTIDE:'''',''''''''''''''''''
' ''' ''''''''
580 DATA 'PEPTIDBINDUNG''''
''''''''',''''''''''''' '''''
' '''''' ''',''{'''''''' '' ' '''
''''''''''',hn coo)
 r''''''''
590 DATA 'bz y w
''''' ' '',''''ch +
c''''''''',''''w (
y z''''''''',''''r h
n coo)'''
600 DATA '''''''''''''''''''b''
'''''''''',''''''''''''''''''''
''''''''''',''''''''''''''''''''
''''''''''',''''''''''''''''''''
'o r''''''
610 DATA '''''''''''''''( w
 w'''''' ,''-ho'''''hn c
 ch''''',''(==a''''''bz y
 z y z'''', ''==)''''''''''''ch
 n coo)
620 DATA '+ho''''''''''''w
h'''''''',''''a''''''''''''r,EN
D
630 !AMP
640 DATA ''''''''''''''''''''''
''''''''',''AMP: ADENOSINMONO
PHOSPHAT''''',''''''''''''''''
''''''''' ''', '''' '''''''''''
''nh'''''''''
650 DATA ''''''''''''''''''''''w
 a '''''' ,'''''' '''' ' ''n
c '''''' '' ,'''''''' 'h tk z
y l'' '''''''''''''''''' z kg
c n'''''''
660 DATA ''''''''''''''c i
 w '''''' ,''' o z c
 cxh'' ''',''''i z y
z k''''''',''}oxpxoxch n
 n''''' '''
670 DATA ''''w w a w''''
' '' '' ''',''''o } w o w'''
''''''''' ',''''''''' w y z w''
'''' ' ' ''' ' ,'''' ''w y zw'
'''''''''''
680 DATA ''''''''''c h h c' '''
'' '' ' ' ''''''''',''''w z w wyw'''
''' ''''' '''''',''''h cxc h''
'''''''' '' ,'''''''''' w w'''
'' ' ''''''''
690 DATA '''''''''''ho oh'''''
''''''''',END
700 !ADP
710 DATA ''''''''''''''''''''''
'''''''',''ADP: ADENOSINDIPH
O S P H'AT '''' ;'' '' '''' '''
''''''''''''',''''''''''''''''''
'' ''''''''''
712 !
713 !Achtung!!! Die runden
 Klammern sind in der nae
 chsten Zeile nicht mit
 einzutippen. Dafuer der
 Buchstabe innerhalb zu-
714 !sammen mit der CTRL-Tas
 te.
715 !
720 DATA ''''''''''''''''''''''
'' '''''' ,''''' o} o
 'u' '''''',''''w i
 vl(A)'''''''',''}oxp~o~pxox ch
 (B)(C)(D)'''''''''
730 DATA ''''i w w a w
''''''''''''''''o o} w o
w''''' ''' '''''''''' '''' w y z
 w'''''' '' ''''''' ''' ' wy
```

```
zw ''"''''''
740 DATA ''''''''''''c h h c
'''''''" ,"'''''W''z  w wy
w "''' '' ' ',''''''" '  h cxc
 h''''''''',''''''''''''''w
w '"'" '''''
750 DATA ''' ''''''''''ho oh'
' ''' '''"''' ''''' '''''''
''''''"'','~=ANHYDRIDBINDUN
G'""''''',END
760 !ATP
770 DATA ''''''''''''''''"''''
''''''''','ATP: ADENOSINTRIP
HOSPHAT''''''''' ''''''''
'''' '''''''',''''''''''''' '''
''''''''''';
774 !
775 !Fuer die naechste Zeile
    gilt die gleiche Eingabe
    vorschrift, wie fuer Zei
    le 720
776 !
780 DATA ''''"'' "'" '''''''
''''''''''',"o   o)  o
   'u'''',"'i   w   i
   vl(A)''','}oxp~o~p~o~px
oxch     (B)(C)(D)'''
790 DATA ''''w   i   w   w a
  w'"'','''o}   o   o}  w
 o  w''''  ''''''''''''''''w
 y z w ''"  '''''''''''"' '''
wy    zw"'''
800 DATA ''''''''''''''''c h
 h c '''  "''',''''''''''' wz
w w  ''yw ,' ''''''''''''''h
 cxc h  ' '; ''''''''' '''''
''w w'''''''
810 DATA '''''''''''''''''ho
 oh'''''','''''''''''''''''
''''''''','~=ANHYDRIDBINDUN
G,END
820 !ADENIN&GUANIN
830 DATA 'PURIN-DERIVATE:''
''''''''','BASEN:'''''''
'''''''''',''''''''''''''''
''''''''"'','ADENIN:     G
UANIN:''''''
840 DATA '''''"'''"''''''
''"" ' ' , "'' n h
 o'''''''','"''w a
  i ' ' ' ''''' c
   c''''''''
850 DATA ''''k z        y
 z'"''''','''n   cxn     hn
   cxn'''','''w   i  l
w   i  l''','''c   c   c
 c   c   c''
860 DATA '''wl y z yw    y l
 y z yw",'''h n   n h  hn
 n   n h' ''''' "' ' h    a
      h'''',END
870 !ADENOSIN&GUANOSIN
880 DATA 'PURIN-DERIVATE:''
''''''''','NUCLEOSIDE:'"'''
''''''''''',''''''''''''"''
''''''''''','ADENOSIN:'''GUA
NOSIN:''''''
890 DATA ''''''''"'"'''''''
''''''''',''''nh'''''''''''
'o'' ' ' ' ' "'' w a'''''''''
''  i'''''''',''''c''''''''''
'''c'''''''''
900 DATA ''''k z''''' '''''''y
 z"''''' ''''n   cxn     h n
   cxn'''','''w   i  l
w   i  l''','''c   c   c
 c   c   c''
910 DATA '''wl y z yw    y l
 y z yw'','''h n   n h  hn
 n   n h' ' '" ' ' '" w
     w'''',''''''RIBOSE
     RIBOSE'',END
920 !CYTOSIN&THYMIN
930 DATA 'PURIMIDIN-DERIVATE
:'''''''','BASEN:'''''''''''
'''''''''',''''''''''''''''
''''''''''  ,"CYTOSIN:
THYMIN:''''
940 DATA '''''"'''''''"''''
''''  ""  ' ''''n h
 o "'' ''' '''''w a
  i "" ' ' ' ''''' c   h
   c   ch"''
950 DATA ''''k z y        y
 z y  b'','''n   c       hn
   c'"''','''w   i
w   i''''','''c   c
 c   c "'''
960 DATA ''k z y z        k z
 y z''''','o   n   h    o
 n   h'''' ,''''h
  h''''''',END
970 !CYTIDIN&THYMIDIN
980 DATA 'PURIMIDIN-DERIVATE
:''''''''','NUCLEOSIDE:'''''
''''''''''','CYTIDIN:      TH
YMIDIN:'"',''"''"""''
'''''''''''''
990 DATA '''''nh           o
''''''''','''''w a
i'' ' ' ' ''''' c   h
 c   ch''''',''''k z y
 y z y  b'''
1000 DATA '''n   c       hn
  c"'''''',''w   i       w
   i"''''','''c   c
c   c''''''','''k z y z
k z y z''''''
1010 DATA 'o   n   h    o
n   h'''',''''''w
 w'''''''',''RIBOSE      R
```

```
IBOSE"'''' '',END
1020 !ALPHA&BETA-D-GLUCOSE
1030 DATA 'MONOSACCHARIDE:"
''''''''''',''' ' ''''''''''''
'' ' ''''''','d-D-GLUCOSE:  e
-D-GLUCOSE:','''''''''''''''''
''''''''''''''
1040 DATA '''hoch           h
och'''' '''';'''w a
 wa' '''' ','''cxxo
  cxxo''''',''h y    z h
 h y    z oh'
1050 DATA ''wy       zw     wy
      zw'','' c       c     c
          c'',''wz oh h yw
wz oh h yw''
1060 DATA 'ho zw  wy oh   ho
zw  wy h'','''' cxxc
  cxxc''''',''''''w   w
   w  w ' '' ' ; ''' h   oh
    h  oh'''',END
1070 !BETA-D-GALACTOSE&FRUKT
OSE
1080 DATA 'MONOSACCHARIDE:"
''' '''' ' ' ,'''''''''''' ''' '
''''''''''-,e-D-GALACTOSE: e
-D-FRUKTOSE:;''''''''''''
''''''''''''''
1090 DATA '''hoch
  o ' ' ' ' '''',' ' w   a       hco
h y z  oh'','''cxxo         a
w y    z w''',ho y     z oh
 wy       zw'''
1100 DATA 'wy         zw       c
h  ho c''','c          c      w
zw    wyw''','wz oh h yw
h cxxxc choh,'h zw   wy h
   w   w   a'
1110 DATA ''''cxxc            h
o    h'' ' '' ,'' w   w ''''' ''
''''''''''''' ,'' ' h   oh''''''
''''''''''''',END
1120 !BETA-D-RIBOSE
1130 DATA 'MONOSACCHARIDE:"
'''''''''' ,''''''''''' ' ' '''
''''''''''-,'e-D-RIBOSE:''''
''''''''''''',''''''''''''''''
''''''''''''''
1140 DATA ''''''''''''o'''''' '
''''''''''',''hcoh y z  oh'
'''''''''''';''''aw y    z w'
''' ' ' ' ''' , ' ' 'w y      zw
''''''''''''''
1150 DATA ''''''c h    h c'''
''''' ''' , '''w z w    wyw''
'''''''''' , ' '''' h cxxxc h'
'' '''' ''''' '''',''' 'w   w''
''''''''''''''
1160 DATA ''''''''ho    oh''''
''''''''''',END
1170 !SACCHAROSE
```

```
1180 DATA 'DI-SACCHARIDE:''
'' '' ''' ',''' ''''''''''' '''
''''''''''','SACCHAROSE:' ''
''''''''''''',''''''''''''''''
''' '''''''''
1190 DATA ''hoch''''''''''''
''''''''''',''''w a ''''''''''
'''''';'''' cx''x  o  ''''''''
''''''''''''' ,'h y'''' z'h''''
''' ' ' ' ' '''''
1200 DATA 'wy          z w'''''
''''''''''', 'c          cd     o
h  h'''''','wz oh h y z
w   w '''',h o zw   wy    o
 cxxxc choh''
1210 DATA ''''cxxc       z yw
  wzw a''''';'' 'w   w       ec h
  ho c ' ' ;''''h ho         wz
     y  w' ' '''''''',''''''  'w
 z    y w''''''
1220 DATA ''''''''''''''hoch z
 y  h''''   ''''''''''''''''' ' a
 o ''''''''''',END
1230 !MALTOSE
1240 DATA 'DI-SACCHARIDE:''
''''''''''', ' ' '''' ' ' ' '' ''
'''''''''','MALTOSE:'''''''
''''''''''''' ,'''''''''''''''''
''''''''''''''
1250 DATA ''hoch            hoc
h''''' ' ' ' ' ''''w a
w a''''''' ,'''c x x o
 cxxo''''''','h y     z h    h
 y     z h''''
1260 DATA 'wy         zw    wy
     zw'''','c          cd   c
         c'''','wz oh h y z y z
 oh h yw''',ho zw    wy     o
 zw   wy oh'''
1270 DATA ''''cxxc              c
xx c ' ' ' ' ' ' ,'''' w   w
w   w ''' ' ' '''''h   oh
 h   oh''''''',END
1280 !MONOACYLGLYCERIN
1290 DATA 'FETTE:'''''''''''
'' ' ' ' '''' ,''''''''''''''''''
''''''''''' ,'MONOACYLGLYCERI
N: '''''' '  ' ''''';''''''''  ''
'''''''''''''
1300 DATA ''''''''''o hoxch'''
'''''''''''',''''''''''i     w a'
'''' ' ' ''' ',hcx(ch)xcxoxxch'
''''''''''''-,'b     af      w'
''' ' ' '''''''''
1310 DATA ''''''''''''hoxch'''
''''''''''''',''''''''''''''''''a,
END
1320 !DIACYLGLYCERIN
1330 DATA 'FETTE:''''''''''
''''''''''''',''''''''''''''''' ,
```

```
''''''''''''''DIACYLGLYCERIN'
'''''''''''' ''''''''''''''''
' '  ''''''  '  '
1340 DATA '''' ''''d''''''
'''' '''''',''''''''''i''' ''
'''''' '''''','hcx(ch)xcxoxch'
''''''''''''','''b    af    w
a''''''''''''''
1350 DATA ''''''''''o   w '''
''''''''''''' ''','''''''i  w'''
'''''''''''''"hcx(ch)xcxoxch'
'''''' ' '''',''' b    af    w'
''''''''''''''
1360 DATA '''''''''''''w''''
''''''''''' ''','''''''''  'w'''
''''''''''''','''''' ''''''w ''
''''''''''''''','''h''b''x  c  h
''''''''''''''
1370 DATA '''''''''''''''a''
'''''''''''',END
1380 !TRIACYLGLYCERIN
1390 DATA 'FETTE:'''''''''''
'''''''''''''','''''''''''''''''''
''''''''''','TRIACYLGLYCERIN
''''''''''''''','''''''''''''''''
''''''''''''''
1400 DATA ''''''''''o''''''''
''''''''''''','''''''''  i'''''''
''''''''','hcx(ch)xcxoxch'
''''''''''''','''b    af    w
a'''''''''''''
1410 DATA ''''''''''o   w ''''
''''  '''' ,'''''''''  'i   w'''
''''''''''','hcx(ch)xcxoxch'
''''''''''''','''b    af    w'
''''''''''''''
1420 DATA ''''''''''''''''w'''
'''' '' ' ' ,''''''' d''  ' ' w'''
'  '''''' ' ,''''''''i'  ''w'  '
'''''''''''''',"hcx(ch)xcxoxch
''''''''''''''
1430 DATA ''b''''af'''''''a''
'''''''''''',END
1440 DIM DAT$(68),BIL$(24)::
 DISPLAY AT(13,1)BEEP:"BITTE
 HABEN SIE ETWAS         GEDUL
D, DIE DATENSAETZE       WERDE
N AUSGELESEN."
1450 I5=0 :: FOR I1=0 TO 14
STEP 3 :: I5=I5+1 :: CALL LO
ESCHEN(BIL$())
1460 I=0 :: ON I5 GOSUB 1890
,1910,1930,1950,1970
1470 I=I+1 :: READ A$ :: IF
A$<>"STOP" THEN BIL$(I)=A$ :
: GOTO 1470 ELSE 1480
1480 RESTORE 1850 :: I=7
1490 I=I+1 :: READ A$ :: IF
A$<>"SUB" THEN BIL$(I)=A$ ::
 GOTO 1490 ELSE 1500
1500 I=11 :: ON I5 GOSUB 190
```

```
0,1920,1940,1960,1980
1510 I=I+1 :: READ A$ :: IF
A$<>"END" THEN BIL$(I)=A$::
 GOTO 1510 ELSE 1520
1520 ZW1$,ZW2$,ZW3$=""
1530 FOR I2=1 TO 8 :: ZW1$=Z
W1$&BIL$(I2):: ZW2$=ZW2$&BIL
$(I2+8):: ZW3$=ZW3$&BIL$(I2+
16):: NEXT I2
1540 DAT$(I1)=ZW1$ :: DAT$(I
1+1)=ZW2$ :: DAT$(I1+2)=ZW3$
1550 PRINT #1:DAT$(I1):: PRI
NT #1:DAT$(I1+1):: PRINT #1:
DAT$(I1+2)
1560 DAT$(I1),DAT$(I1+1),DAT
$(I1+2)=""
1570 DISPLAY AT(20,26):I5 ::
 NEXT I1
1580 I1=0 :: FOR I2=15 TO 66
 STEP 3 :: I1=I1+1 :: CALL L
OESCHEN(BIL$())
1590 I=0
1600 IF I1>7 THEN 1620
1610 ON I1 GOSUB 1870,1880,1
990,2000,2010,2020,2030 :: G
OTO 1650
1620 IF I1>15 THEN 1640
1630 ON I1-7 GOSUB 2040,2050
,2060,2070,2080,2090,2100,21
10 :: GOTO 1650
1640 ON I1-15 GOSUB 2120,213
0,2140
1650 I=I+1 :: READ A$ :: IF
A$<>"END" THEN BIL$(I)=A$ ::
 GOTO 1650 ELSE 1660
1660 ZW1$,ZW2$,ZW3$=""
1670 FOR I3=1 TO 8 :: ZW1$=Z
W1$&BIL$(I3):: ZW2$=ZW2$&BIL
$(I3+8):: ZW3$=ZW3$&BIL$(I3+
16): NEXT I3
1680 DAT$(I2)=ZW1$ :: DAT$(I
2+1)=ZW2$ :: DAT$(I2+2)=ZW3$
1690 PRINT #1:DAT$(I2):: PRI
NT #1:DAT$(I2+1):: PRINT #1:
DAT$(I2+2)
1700 DAT$(I2),DAT$(I2+1),DAT
$(I2+2)=""
1710 DISPLAY AT(20,26):I2 ::
 NEXT I2
1720 DISPLAY AT(10,1):"DAS A
USLESEN DER DATENSAETZEIST B
EENDET !"
1730 CALL SAY("THE+WORK+IS+D
ONE"):: CLOSE #1 :: STOP
1740 DISPLAY AT(13,1):"BITTE
 LEGEN SIE EINE        DISKE
TTE EIN, AUF DIE DIE   DATEN
 GESPEICHERT WERDEN"
1750 DISPLAY AT(16,1):"SOLLE
N."
```

```
1760 CALL SAY("#PLEASE##PUT#
#IN##A1##DISKETTE#,#I##WANT#
#TO##SAVE##THE1##DATA#")
1770 DISPLAY AT(20,1):" BIT
TE TASTE DRUECKEN !!!" :: CA
LL SAY("#PRESS##A1##KEY#")
1780 DISPLAY AT(20,1):" BIT
TE TASTE DRUECKEN !!!" :: CA
LL KEY(0,K,S):: IF S=0 THEN
DISPLAY AT(20,1):"" :: GOTO
1780 ELSE 1790
1790 CALL CLEAR :: DISPLAY A
T(10,1)BEEP:"LAUFWERKNUMMER
(1-4):" :: CALL SAY("#WHICH#
#DISKETTE##MEMORY#")
1800 ACCEPT AT(10,23)VALIDAT
E("1234")SIZE(1):NM
1810 FILE$="DSK"&STR$(NM)&".
BIODAT"
1820 CALL SAY("#PLEASE##WAIT
#,I+AM+WORKING")
1830 OPEN #1:FILE$,RELATIVE,
INTERNAL,OUTPUT,FIXED 255
1840 RETURN
1850 !HAUPTTEIL AMINOSAEUREN
1860 DATA '''coo}
coo}'''''','{ w            {
 w'"" ' ';'hnxch           h
nxcxh'''''''','b w
 b w "''''''",SUB
1870 RESTORE 250 :: RETURN
1880 RESTORE 300 :: RETURN
1890 RESTORE 350 :: RETURN
1900 RESTORE 370 :: RETURN
1910 RESTORE 390 :: RETURN
1920 RESTORE 420 :: RETURN
1930 RESTORE 430 :: RETURN
1940 RESTORE 460 :: RETURN
1950 RESTORE 480 :: RETURN
1960 RESTORE 500 :: RETURN
1970 RESTORE 520 :: RETURN
1980 RESTORE 540 :: RETURN
1990 RESTORE 560 :: RETURN
2000 RESTORE 630 :: RETURN
2010 RESTORE 700 :: RETURN
2020 RESTORE 760 :: RETURN
2030 RESTORE 820 :: RETURN
2040 RESTORE 870 :: RETURN
2050 RESTORE 920 :: RETURN
2060 RESTORE 970 :: RETURN
2070 RESTORE 1020 :: RETURN
2080 RESTORE 1070 :: RETURN
2090 RESTORE 1120 :: RETURN
2100 RESTORE 1170 :: RETURN
2110 RESTORE 1230 :: RETURN
2120 RESTORE 1280 :: RETURN
2130 RESTORE 1320 :: RETURN
2140 RESTORE 1380 :: RETURN
2150 SUB LOESCHEN(BIL$())
2160 FOR I=1 TO 24 :: BIL$(I
)=RPT$("'",28): NEXT I :: S
UBEND
```

LISTINGS

```
100 !***********************
*                      *
*      BIOCHEMIE       *
*      HAUPTPROGRAMM   *
*                      *
110 !*   V E R S I O N 1.0*
*                      *
*(C) 1986 BY DIRK      *
*            JUNGHANS  *
*            (DJIS)    *
120 !*                     *
*SYSTEM:TI99/4A        *
*EXTENDED BASIC        *
*BIOCHEMIE GENERATOR   *
*32KRAM ERWEITERUNG    *
130 !*DSK1 OPT. DSK2       *
*SPEECH SYNTHESIZER    *
*                      *
*bei fragen:           *
*dirk junghans         *
140 !*am fort-biehler 9    *
*6503 mainz-kastel     *
************************

145 !

150 DATA 007C02111161221,21
2224280B102040,000408000080
808,1810101010103010,0008142
222224242,82848484C4442438,0
006090911103040
160 DATA 4040404021111 20C,00
081010101010 1C,1311312121212
222,000F8888889890 90,1F1110 1
010101E01,00004C60504040404,0
4840808080808 08
170 DATA 0000216 0A1A12121,63
42424242428682,008F0808 08181
010,1F111010101 01E01,0000C,0
08,007C2221212 12224,007E8139
11113901,28302040 8
180 DATA 0D110985592112 0C
181 ON WARNING NEXT
190 CALL CLEAR :: PRINT TAB(
10);"BITTE WARTEN" :: CALL S
AY("PLEASE+WAIT"):: FOR I=1
TO 24 :: READ A$ :: CALL CHA
R(I+96,A$): NEXT I
191 CALL SPGET("B",A$):: CAL
L SPGET("O",B$):: CALL SPGET
("SHAPE",C$):: CALL SPGET("E
IGHTY",D$):: CALL SPGET("ME"
,E$)
192 AN$=CHR$(96)&CHR$(0)
193 BSP2$=AN$&CHR$(21)&SEG$(
A$,4,21)&AN$&CHR$(26)&SEG$(B
$,4,26)&AN$&CHR$(18)&SEG$(C$
,4,18)&AN$&CHR$(28)&SEG$(D$,
4,28)
194 BSP2$=BSP2$&AN$&CHR$(26)
&SEG$(E$,4,26)
195 CALL SPGET("WILL",A$) ::
```

```
CALL SPGET("COURSE",B$):: CA
LL SPGET("ME",C$):: CALL SPG
ET("N",D$)
196 BSP1$=AN$&CHR$(56)&SEG$(
A$,4,56)&AN$&CHR$(22)&SEG$(B
$,4,22)&AN$&CHR$(17)&SEG$(C$
,4,17)&AN$&CHR$(46)&SEG$(D$,
4,46)
22Ø GOSUB 277
23Ø DISPLAY AT(7,9):"acegikm
oqs" :: DISPLAY AT(8,9):"bdf
hjlnprt" :: DISPLAY AT(2,2):
"uv" :: DISPLAY AT(3,2):"wx"
24Ø DISPLAY AT(19,1):"DIESES
 PROGRAMM IST           KATAJO
N NAZARI             (GEB.
Ø1.1Ø.1986) GEWIDMET. WEITER
 DANKE ICH HERRN UND"
25Ø DISPLAY AT(23,1):"FRAU D
R. NAZARI." :: DISPLAY AT(15
,1):"    (C) 1986 BY DIRK JUN
GHANS            (DJIS)"
26Ø DISPLAY AT(2,18):"PROGRA
MMED" :: DISPLAY AT(3,18):"B
Y DJIS" :: CALL SAY(,BSP1$,"
,BY,",BSP2$):: CALL SAY("PRE
SS+A1+KEY+TO+START",BSP2$)
27Ø CALL KEY(Ø,K,S):: IF S=Ø
 THEN 27Ø ELSE CALL CLEAR ::
 PRINT TAB(1Ø);"BITTE WARTEN
" :: CALL SAY("PLEASE+WAIT")
271 CALL CHARSET :: FOR I=11
5 TO 99 STEP -1 :: DISPLAY A
T(23,26):I-73 :: CALL CHARPA
T(I-32,A$):: CALL CHAR(I,A$)
:: NEXT I :: GOSUB 277
272 CALL CHAR(119,RPT$("1Ø",
8),12Ø,"ØØØØØØFF",121,"Ø1Ø2Ø
4Ø81Ø2Ø4Ø8Ø",122,"8Ø4Ø2Ø1ØØ8
Ø4Ø2Ø1")
273 CALL CHAR(97,"6Ø9Ø2Ø4ØFØ
",98,"6Ø9Ø3Ø9Ø6",1ØØ,"ØØ3ØCA
844A3",1Ø1,"ØØ18444844484Ø4"
,1Ø3,"8",116,"ØØØØØØØØØØØØØØØ
Ø1")
274 CALL CHAR(1Ø2,"AØ5Ø5Ø5Ø"
,1Ø5,RPT$("14",8),1Ø6,"ØØØØF
FØØFF",1Ø7,"Ø2Ø5ØA14285ØAØ4"
,1Ø8,"4ØAØ5Ø2814ØAØ5Ø2")
275 CALL CHAR(129,"848F4ØA95
D4BØ9Ø45Ø6Ø4Ø2Ø19ØDØBØ928282
82Ø528BØ6Ø2Ø414281Ø6Ø4ØCØ4Ø"
):: CALL CHAR(117,"ØØØØØ5Ø7
Ø7659810ØØØØØØØ9ØDØB192Ø")
276 CALL CHAR(96,"ØØØØØØØ9ØD
ØBØ9",124,"ØØØ1Ø2Ø4885Ø2Ø28"
,39,RPT$("Ø",16)):: GOTO 28Ø
277 CALL SCREEN(15):: FOR I=
Ø TO 8 :: CALL COLOR(I,5,15)
:: NEXT I :: FOR I=9 TO 14 :
: CALL COLOR(I,2,15):: NEXT
I
278 FOR I=9Ø TO 65 STEP -1 :
: DISPLAY AT(23,26):I-65 ::
CALL CHARPAT(I,A$):: CALL CH
AR(I,"ØØ"&SEG$(A$,1,4)&SEG$(
A$,7,1Ø)):: NEXT I
279 CALL CHAR(67,"ØØØØ384Ø4Ø
4Ø4Ø38",75,"ØØØØ485Ø6Ø5Ø4844
"):: RETURN
28Ø CALL CHAR(123,"ØØØØ1Ø381
Ø",125,"ØØØØØØ38",64,"ØØØØØØ
ØØØØ3Ø1Ø2Ø"):: ST$=RPT$("*",
28)
281 GOSUB 195ØØ
289 DISPLAY AT(1,1)ERASE ALL
:"  BIOCHEMIE (C) 1986 BY
              DIRK JUNGHANS
   "&ST$
29Ø DISPLAY AT(5,1):"[1] AMI
NOSAEUREN": :"[2] PEPTIDE":
:"[3] NUCLEOTIDE": :"[4] PRO
TEINE": :"[5] NUCLEINBASEN N
UCLEOSIDE"
3ØØ DISPLAY AT(15,1):"[6] MO
NOSACCHARIDE": :"[7] DI-SACC
HARIDE": :"[8] FETTE": :"[9]
 ALLGEMEINE ABKUERZUNGEN": :
"[Ø] ENDE"
3Ø1 CALL SAY("#PRESS##A1##NU
MBER#")
31Ø CALL KEY(Ø,K,S):: IF K<4
8 OR K>57 THEN 31Ø ELSE 32Ø
32Ø ON K-47 GOTO 33Ø,5ØØ,25Ø
Ø,45ØØ,65ØØ,85ØØ,1Ø5ØØ,125ØØ
,145ØØ,4ØØ
33Ø PRINT "BYE !": :"DJIS" :
: CALL SAY("GOODBYE"):: CLOS
E #1 :: STOP
4ØØ DISPLAY ERASE ALL BEEP A
T(1,1):"[1] VORWAERTS [2] NO
CHMAL    [Ø] ZURRUECK ZUM HAU
PTMENUE":ST$
41Ø RESTORE 15ØØØ :: I=Ø ::
I1=3
42Ø I=I+1 :: I1=I1+1 :: READ
 A$ :: IF A$="BREAK" THEN 43
Ø ELSE IF A$="STOP" THEN 46Ø
 ELSE DISPLAY AT(I1,1):A$ ::
 GOTO 42Ø
43Ø I1=3 :: CALL SAY("KEY")
44Ø CALL KEY(Ø,K,S):: IF K<4
8 OR K>5Ø THEN 44Ø ELSE ON K
-47 GOTO 289,45Ø,4ØØ
45Ø CALL HCHAR(4,1,32,64Ø)::
 GOTO 42Ø
46Ø DISPLAY AT(1,1)BEEP:"[1]
 NOCHMAL":"[Ø] ZURRUECK ZUM
HAUPTMENUE" :: CALL SAY("PRE
SS+A1+NUMBER")
47Ø CALL KEY(Ø,K,S):: IF K=4
```

```
8 THEN 289 ELSE IF K=49 THEN
 400 ELSE 470
500 !AMINOSAEUREN
510 DISPLAY ERASE ALL BEEP A
T(1,1):"          AMINOSAEUREN
     "&ST$
520 DISPLAY AT(4,1):"[1] AMI
NOSAEUREN ALLGEMEIN": :"[2]
AMINOSAEUREN MIT EINER
REINEN C-H SEITENKETTE."
530 DISPLAY AT(9,1):"[3] AMI
NOSAEUREN MIT NICHT-     ION
ISIERTEN ABER POO-       LAR
EN GRUPPEN IN DER        SEI
TENKETTE."
540 DISPLAY AT(14,1):"[4] SA
URE AMINOSAEUREN.": :"[5] BA
SISCHE AMINOSAEUREN.": :"[0]
 ZURRUECK ZUM HAUPTMENUE."
560 CALL SAY("PRESS+A1+NUMBE
R")
570 CALL KEY(0,K,S):: IF K<4
8 OR K>54 THEN 570 ELSE ON K
-47 GOTO 289,600,700,800,900
,1000
600 A=15 :: GOSUB 19000
610 GOTO 1010
700 A=18 :: GOSUB 19000 :: C
ALL AUSGABE(A$,B$,C$)
710 CALL KEY(0,K,S):: IF S=0
 THEN 710
720 A=0 :: GOSUB 19000
730 GOTO 1010
800 A=3 :: GOSUB 19000 :: CA
LL AUSGABE(A$,B$,C$)
810 CALL KEY(0,K,S):: IF S=0
 THEN 810
820 A=6 :: GOSUB 19000
830 GOTO 1010
900 A=9 :: GOSUB 19000
910 GOTO 1010
1000 A=12 :: GOSUB 19000
1010 CALL AUSGABE(A$,B$,C$):
: GOTO 500
2500 !PEPTIDE
2510 DISPLAY ERASE ALL BEEP
AT(1,1):"              PEPTIDE
      "&ST$
2520 DISPLAY AT(4,1):"[1] BA
UPRINZIP DER PEPTIDE/     PE
PTIDBINDUNG.": :"[0] ZURRUEC
K ZUM HAUPTMENUE."
2530 CALL SAY("PRESS+A1+NUMB
ER")
2540 CALL KEY(0,K,S):: IF K<
48 OR K>49 THEN 2540 ELSE ON
 K-47 GOTO 289,2600
2600 A=21 :: GOSUB 19000
2610 CALL AUSGABE(A$,B$,C$):
: GOTO 2500
4500 !NUCLEOTIDE
4510 DISPLAY ERASE ALL BEEP
AT(1,1):"          NUCLEOTIDE
      "&ST$
4520 DISPLAY AT(4,1):"[1] AM
P": :"[2] ADP": :"[3] ATP":
:"[0] ZURRUECK ZUM HAUPTMENU
E."
4530 CALL SAY("PRESS+A1+NUMB
ER")
4540 CALL KEY(0,K,S):: IF K<
48 OR K>51 THEN 4540 ELSE ON
 K-47 GOTO 289,4600,4700,480
0
4600 A=24 :: GOSUB 19000
4610 GOTO 4810
4700 A=27 :: GOSUB 19000
4710 GOTO 4810
4800 A=30 :: GOSUB 19000
4810 CALL AUSGABE(A$,B$,C$):
: GOTO 4500
6500 DISPLAY AT(1,1)ERASE AL
L BEEP:"[1] WEITERBLAETTERN"
:"[0] ZURRUECK ZUM HAUPTMENU
E":ST$
6510 RESTORE 16000 :: I=0 ::
 I1=3
6520 I=I+1 :: I1=I1+1 :: REA
D A$ :: IF A$="BREAK" THEN 6
530 ELSE IF A$="STOP" THEN 6
550 ELSE DISPLAY AT(I1,1):A$
 :: GOTO 6520
6530 CALL SAY("KEY")
6540 CALL KEY(0,K,S):: IF K<
48 OR K>49 THEN 6540 ELSE ON
 K-47 GOTO 289,6560
6550 DISPLAY AT(1,1)BEEP:""
6551 CALL KEY(0,K,S):: IF K=
48 THEN 289 ELSE 6551
6560 I1=3 :: CALL HCHAR(4,1,
32,640):: GOTO 6520
8500 !NUCLEINBASEN NUCLEOSID
E
8510 DISPLAY ERASE ALL BEEP
AT(1,1):" NUCLEINBASEN NUCL
EOSIDE    "&ST$
8520 DISPLAY AT(4,1):"[1] PU
RIN DERIVATE :BASEN": :"[2]
PURIN DERIVATE :NUCLEO-
SIDE": :"[3] PURIMIDIN DERIV
ATE :BA-     SEN"
8530 DISPLAY AT(12,1):"[4] P
URIMIDIN DERIVATE :NU-     C
LEOSIDE": :"[0] ZURRUECK ZUM
 HAUPTMENUE" :: CALL SAY("PR
ESS+A1+NUMBER")
8540 CALL KEY(0,K,S):: IF K<
48 OR K>52 THEN 8540 ELSE ON
 K-47 GOTO 289,8600,8700,880
0,8900
8600 A=33 :: GOSUB 19000
```

LISTINGS

```
8610 GOTO 8910
8700 A=36 :: GOSUB 19000
8710 GOTO 8910
8800 A=39 :: GOSUB 19000
8810 GOTO 8910
8900 A=42 :: GOSUB 19000
8910 CALL AUSGABE(A$,B$,C$):
: GOTO 8500
10500 !MONOSACCHARIDE
10510 DISPLAY ERASE ALL BEEP
 AT(1,1):"          MONOSACCHA
RIDE        "&ST$
10520 DISPLAY AT(4,1):"[1] G
LUCOSEN": :"[2] GALACTOSE/FR
UKTOSE": :"[3] RIBOSE": :"[0
] ZURRUECK ZUM HAUPTMENUE"
10530 CALL SAY("PRESS+A1+NUM
BER")
10540 CALL KEY(0,K,S):: IF K
<48 OR K>51 THEN 10540 ELSE
ON K-47 GOTO 289,10600,10700
,10800
10600 A=45 :: GOSUB 19000
10610 GOTO 10810
10700 A=48 :: GOSUB 19000
10710 GOTO 10810
10800 A=51 :: GOSUB 19000
10810 CALL AUSGABE(A$,B$,C$)
:: GOTO 10500
12500 !DISACCHARIDE
12510 DISPLAY ERASE ALL BEEP
 AT(1,1):"          DI-SACCHAR
IDE         "&ST$
12520 DISPLAY AT(4,1):"[1] S
ACCHAROSE": :"[2] MALTOSE":
:"[0] ZURRUECK ZUM HAUPTMENU
E"
12530 CALL SAY("PRESS+A1+NUM
BER")
12540 CALL KEY(0,K,S):: IF K
<48 OR K>50 THEN 12540 ELSE
ON K-47 GOTO 289,12600,12700
12600 A=54 :: GOSUB 19000
12610 GOTO 12710
12700 A=57 :: GOSUB 19000
12710 CALL AUSGABE(A$,B$,C$)
:: GOTO 12500
14500 !FETTE
14510 DISPLAY ERASE ALL BEEP
 AT(1,1):"              LIPID
E           "&ST$
14520 DISPLAY AT(4,1):"[1] M
ONOACYLGLYCERIN": :"[2] DIAC
YLGLYCERIN": :"[3] TRIACYLGL
YCERIN": :"[0] ZURRUECK ZUM
HAUPTMENUE"
14530 CALL SAY("PRESS+A1+NUM
BER")
14540 CALL KEY(0,K,S):: IF K
<48 OR K>51 THEN 14540 ELSE
ON K-47 GOTO 289,14600,14700
,14800
14600 A=60 :: GOSUB 19000
14610 GOTO 14810
14700 A=63 :: GOSUB 19000
14710 GOTO 14810
14800 A=66 :: GOSUB 19000
14810 CALL AUSGABE(A$,B$,C$)
:: GOTO 14500
15000 !ABKUERZUNGEN
15010 DATA A(ADO).:ADENOSIN,
ADP....:ADENOSINDIPHOSPHAT,A
LA....:ALANIN,AMP....:ADENOS
INMONOPHOSPHAT
15020 DATA ARG....:ARGININ,A
SN....:ASPARAGIN,ASP....:ASP
ARAGINSAEURE,ATP....:ADENOSI
NTRIPHOSPHAT,C(CYD).:CYTIDIN
,CDP...:CYTIDINPHOSPHAT
15030 DATA CMP....:CYTIDINMO
NOPHOSPHAT,CTP....:CYTIDINTR
IPHOSPHAT,CYS....:CYSTEIN,DN
A....:DESOXYRIBONUCLEIN-,'''
'''''SAEURE
15040 DATA FAD....:FLAVIN-AD
ENIN-DINU-,'''''''CLEOTID,F
D.....:FERREDOXIN,FMN....:FL
AVINMONONUCLEOTID,BREAK
15050 DATA FMET...:FORMYLMET
HION,FP.....:FLAVOPROTEIN,FR
U....:FRUCTOSE,G(GUO).:GUANO
SIN,GAL....:GALACTOSE,GDP...
.:GUANOSINDIPHOSPHAT
15060 DATA GLC....:GLUCOSE,G
LN....:GLUTAMIN,GLU....:GLUT
AMINSAEURE,GLY....:GLYCIN,GM
P....:GUANOSINMONOPHOS-,''''
''''PHAT
15070 DATA GTP....:GUANOSINT
RIPHOSPHAT,HB.....:HAEMOGLOB
IN,HIS....:HISTIDIN,HYP....:
HYDROXYPROLIN,IDP....:INOSIN
DIPHOSPHAT,BREAK
15080 DATA ILE....:ISOLEUCIN
,IMP....:INOSINMONOPHOSPHAT,
ITP....:INOSINTRIPHOSPHAT,LE
U....:LEUCIN,LYS....:LYSIN,M
AN.....:MANNOSE
15090 DATA MET....:METHION,N
AD....:NICOTINAMID-ADENIN-,'
'''''''DINUCLEOTID,NADH...:R
EDUZIERTES NAD,NADP...:NICOT
INAMID-ADENIN-
15100 DATA '''''''DINUCLEOT
ID-PHOSPHAT,NADPH..:REDUZIER
TES NADP,PHE....:PHENYLALANI
N,PRO....:PROLIN,Q......:UBI
CHINON
15110 DATA RIB....:RIBOSE,RN
A.....:RIBONUCLEINSAEURE,BREA
K
```

LISTINGS

```
15120 DATA RUL.....:RIBULOSE,
SER....:SERIN,T......:THYMIN
,THR....:THREONIN,THX....:TH
YROXIN,TRP....:TRYPTOPHAN,TY
R....:TYROSIN
15130 DATA U......:URIDIN,UD
P....:URIDINPHOSPHAT,UMP....
:URIDINMONOPHOSPHAT,UTP....:
URIDINTRIPHOSPIHAT,VAL....:VA
LIN
15140 DATA XUL....:XYLULOSE,
XYL....:XYLOSE,************
****************,STOP
16000 DATA PROTEINE: PROTEIN
E (EIWEISS-,STOFFE) SIND DIE
 HAUPTBE-,STANDTEILE DES CYT
OPLASMAS.,ES SIND MAKROMOLEK
UEHLE@ DIE
16010 DATA DURCH VERKNUEPFUN
G VON,AMINOSAEUIREN ENTSTEHEN
.,DIE VERSCHIEDENEN AMINO-,S
AEUREN SIND DURCH PEPTID-,BI
NDUNGEN MITEINANDER VER-
16020 DATA BUNDEN (SIEHE PEP
TIDE).,AB EINER ANREIHUNG VO
N ETWA,HUNDERT BIS TAUSEND A
MINO-,SAEURERESTEN@ SPRICHT
MAN,VON PROTEINEN.
16030 DATA MAN UNTERSCHEIDET
 DIE PRO-,TEINE IN ZWEI GRUP
PEN@ DIE,SICH DURCH IHRE CHE
M. EIGEN-,SCHAFTEN VONEINAND
ER ABGREN-,ZEN.,BREAK
16040 DATA DIESE ZWEI GRUPPE
N SIND DIE,SKLEROPROTEINE UN
D DIE,GLOBULAEREN- ODER SPAE
ROPR-,TEINE.
16050 DATA IN DEN PROTEINEN
TRETEN 20,VERSCHIEDENE AMINO
SAEREN,AUF. DIE VERSCHIEDENE
N PRO-,TEINE UNTERSCHEIDEN S
ICH,DURCH DIE ANZAHL UND
16060 DATA REIHENFOLGE (SEQU
ENZ) DER,VERKNUEPFTEN AMINOS
AEUREN.,BREAK
16070 DATA PRIMAERSTRUKTUR:
DIE REIHEN-,FOLGE DER AMINOS
AEUREN IN,EINER POLIPEPTIDKE
TTE HEISST,AMINOSAEURENSEQUE
NZ ODER,PRIMAERSTRUKTUR.
16080 DATA D.H. DIE ANEINAND
ERREIHUNG,EINZELNER AMINOS.
HINTEREI-,NANDER HEISST PRIM
AERSTRUK-,TUR.,SEKUNDAERSTRU
KTUR: PROTEIN-
16090 DATA MOLEKUEHLE HABEN
EINE BE-,STIMMTE RAEUMLICHE
GESTALT;,MAN SAGT@ DIE KETTE
 IST IN,BESTIMMTER WEISE GEF
ALTET.,DIE ART DER FALTUNG
16100DATAISTBEREITS DURCH
 DIE PRI-,MAERSTRUKTUR FESTG
ELEGT.,DIE FALTUNG ERGIBT SI
CH BREAK
16110 DATA NICHT ZUFAELLIG@
SONDERN ALS,KONSEQUENZ BINDE
NDER AB-,SCHNITTE ZWISCHEN D
EN PRO-,TEINEN.,SOWEIT NUR D
IE KETTE SELBST@
16120DATAD.H.DIEATOMEDE
R PEPTID-,BINDUNG@ BETROFFEN
 SIND@,SPRICHT MAN VON SEKUN
-,DAERSTRUKTUR.,TERTIAERSTRU
KTUR: DIE IN
16130 DATA DER SEKUNDERSTR.
ANGEORDNE-,TEN AMINOKETTEN@
DURCHLAUFEN,REGELMAESSIG WIE
DERKEHRENDE,UMKEHRSCHLEIFEN.
 D.H. DIE IN
16140 DATA DER SEKUNSAERSTRU
KTER ANGE-,ORDNETEN AMINOKET
TEN WERDEN,NOCHMAL GEFALTET.
 DIES IST,DANN DIE TERTIAERS
TRUKTUR.
16150 DATA BREAK,HIERBEI TRE
TEN FOLGENDE BIN-,DUNGEN AUF
: H-BRUECKEN-@,DISULFID- UND
 HYDROPHOBE-,BINDUNGEN.,QUAR
TAERSTRUKTUR: WENN SICH
16160 DATA PROTEINE@ DIE IN
DER TERTI-,AERSTRUKTUR VORLI
EGEN@ ZU,HOEHEREN AGGREGATEN
 ZUSAM-,MENSCHLIESSEN@ SO NE
NNT MAN
16170 DATA DIES DIE QUARTAER
STRUKTUR.,*****************
**********,STOP
19000 CALL CLEAR :: INPUT #1
,REC A:A$ :: INPUT #1,REC A+
1:B$ :: INPUT #1,REC A+2:C$
19010 RETURN
19500 DISPLAY AT(8,1)BEEP ER
ASE ALL:"IN WELCHEM LAUFWERK
 BEFINDETSICH DIE DISKETTE M
IT DEN   DATEN DES GENERATOR
PRO-     GRAMMES ?(1-4):"
19510 CALL SAY("#IN##WHICH##
DISKETTE##MEMORY##IS##THE1##
DATA##DISKETTE#")
19520 ACCEPT AT(11,25)SIZE(1
)VALIDATE("1234")BEEP:NM
19530 CALL CLEAR :: FILE$="D
SK"&STR$(NM)&".BIODAT"
19540 OPEN #1:FILE$,RELATIVE
,INTERNAL,INPUT ,FIXED 255
19550 RETURN
20000 SUB RIBOSE(X,Y):: FOR
I=1 TO 3 :: FOR A=1 TO 3 ::
Z=Z+1 :: DISPLAY AT(Y+I,X+I)
:CHR$(Z+127):: NEXT A :: NEX
T I :: SUBEND
```

»

```
20999 SUB AUSGABE(A$,B$,C$)
21000 DISPLAY AT(1,1):A$ ::
DISPLAY AT(9,1):B$ :: DISPLA
Y AT(17,1):C$ :: DISPLAY AT(
24,5):"BITTE TASTE DRUECKEN"
21010 CALL SAY("PRESS+A1+KEY
")
21020 CALL KEY(0,K,S):: IF S
=0 THEN 21020
21030 SUBEND
```

FLÜSSE IN EUROPA

Das Programm „Flüsse in Europa" läuft auf dem TI 99/4A + Extended Basic.
Bei diesem Programm handelt es sich um ein Lernprogramm.
Der Computer hat 26 Flüsse aus Europa gespeichert, aus diesen 26 sucht er sich durch Zufall einen aus.
Nun hat man drei Funktionen zur Verfügung, um herauszufinden, um welchen Fluß es sich handelt.
Wenn man "T" drückt, gibt der Computer einen Tip, den er zufällig auswählt. Er nennt zum Beispiel ein Land, durch das der Fluß fließt, oder er nennt eine größere Stadt, durch die der Fluß fließt usw.
Drückt man jedoch "F", erscheint auf dem Bildschirm:
– DU WILLST ETWAS FRAGEN! –,
nun darf man eine Frage stellen, die aus einem Wort besteht. Hier eine Auflistung der Fragewörter, mit der Beschreibung, was sie bewirken:

- MUENDET: Man erfährt, in welches Meer der Fluß mündet.
- ENTSPRINGT: Der Computer sagt, wo der Fluß entspringt.
- LAND: Hierbei wird ein Land genannt, durch das der Fluß fließt.
- LAENDER: Wenn man dieses Fragewort eingibt, erfährt man, durch wieviele Länder der Fluß fließt.
- STADT: Das gleiche wie bei LAND, nur mit einer größeren Stadt.
- STAEDTE: Das gleiche wie bei LAENDER, nur mit größeren Städten.
- NEBENFLUSS: Das gleiche wie bei LAND, nur mit einem größeren Nebenfluß.
- NEBENFLUESSE: Das gleiche wie bei LAENDER, nur mit größeren Nebenflüssen.
- FLIESSRICHTUNG: Hierbei wird entweder die Richtung genannt, aus der der Fluß kommt, oder die Richtung, in die er fließt.

Wenn man zuviele Fragen stellt, oder sich zuviele Tips geben läßt, sucht der Computer sich einen neuen Fluß aus.
Außer "T" und "F" kann man noch die Taste "R" drücken. Tut man dies, so muß man raten, um welchen Fluß es sich handelt. Rät man falsch, sucht der Computer sich einen neuen Fluß aus und es geht wieder von vorne los. Rät man jedoch richtig, dann kriegt man eine bestimmte Anzahl von Punkten (die Anzahl der Punkte ergibt sich aus der Anzahl der Tips und Fragen) und es geht ebenfalls wieder von vorne los.
Wenn soviele Fragen gestellt wurden, wie man wollte, ist das Programm zu Ende.

```
100 !************************
110 !*  FLUESSE IN EUROPA   *
120 !*     VERSION 1.2      *
130 !************************
140 !*       TI 99/4A       *
150 !*          +           *
160 !*   EXTENDED BASIC     *
170 !************************
180 !*   COPYRIGHT 1985     *
190 !*         BY           *
200 !*  FRANK RADEMACHER    *
210 !*                      *
220 !*                      *
230 !************************
240 DIM FLU$(26),MU$(26),EN$
(26),LA1$(26),LA2$(26),LA3$(
26),LA4$(26),LA5$(26),LA6$(2
6),LA7$(26),LA8$(26),ST1$(26
),ST2$(26),ST3$(26)
250 DIM ST4$(26),FL1$(26),FL
2$(26),NE1$(26),NE2$(26),NE3
$(26),NE4$(26)
260 GOSUB 1700
270 CALL CLEAR :: INPUT "WIE
VIELE FRAGEN :":FRA
280 IF FRA<1 OR FRA>25 THEN
270
290 CALL CLEAR :: RANDOMIZE
:: CALL SCREEN(3)
300 DISPLAY AT(2,7):"FLUESSE
 IN EUROPA"
310 DISPLAY AT(4,3)SIZE(6):"
PUNKTE" :: DISPLAY AT(4,15)S
IZE(5):"FLUSS"
320 DISPLAY AT(7,1):"-------
---------------------"
330 FRG=FRG+1 :: PUNK=10 ::
GOSUB 2130
340 FL=INT(RND*26)+1
350 DISPLAY AT(12,3):"ICH HA
BE EINEN FLUSS," :: DISPLAY
AT(13,3):"DU KANNST RATEN."
360 FOR I=1 TO 250 :: NEXT I
370 CALL HCHAR(12,1,32,96)
380 FG=0 :: DISPLAY AT(20,3)
:"TIP=T" :: DISPLAY AT(21,3)
:"FRAGEN=F" :: DISPLAY AT(22
,3):"RATEN=R"
390 CALL SOUND(100,1000,0)
400 CALL KEY(0,K,S)
410 IF S=0 THEN 400
420 CALL HCHAR(20,1,32,128)
430 DISPLAY AT(14,1):""
440 IF K=ASC("T")THEN 480
450 IF K=ASC("F")THEN 1220
460 IF K=ASC("R")THEN 1110
470 GOTO 400
480 TI=TI+1 :: PUNK=PUNK-1
490 IF PUNK<0 THEN 1510
500 DISPLAY AT(10,10):"TIP N
R."&STR$(TI)
```

```
51Ø ON INT(RND*6)+1 GOTO 52Ø
,55Ø,58Ø,79Ø,92Ø,98Ø
52Ø DISPLAY AT(12,1):"DER GE
SUCHTE FLUSS MUENDET:"
53Ø DISPLAY AT(13,3):MU$(FL)
54Ø GOTO 38Ø
55Ø DISPLAY AT(12,1):"DER FL
USS ENTSPRINGT:"
56Ø DISPLAY AT(13,3):EN$(FL)
57Ø GOTO 38Ø
58Ø ANZ=1
59Ø IF LA2$(FL)="" THEN 66Ø
ELSE ANZ=ANZ+1
6ØØ IF LA3$(FL)="" THEN 66Ø
ELSE ANZ=ANZ+1
61Ø IF LA4$(FL)="" THEN 66Ø
ELSE ANZ=ANZ+1
62Ø IF LA5$(FL)="" THEN 66Ø
ELSE ANZ=ANZ+1
63Ø IF LA6$(FL)="" THEN 66Ø
ELSE ANZ=ANZ+1
64Ø IF LA7$(FL)="" THEN 66Ø
ELSE ANZ=ANZ+1
65Ø IF LA8$(FL)="" THEN 66Ø
ELSE ANZ=ANZ+1
66Ø IF FG=1 THEN 136Ø
67Ø ON INT(RND*ANZ)+1 GOTO 6
8Ø,69Ø,7ØØ,71Ø,72Ø,73Ø,74Ø,7
5Ø
68Ø LA$=LA1$(FL) :: GOTO 76Ø
69Ø LA$=LA2$(FL) :: GOTO 76Ø
7ØØ LA$=LA3$(FL) :: GOTO 76Ø
71Ø LA$=LA4$(FL) :: GOTO 76Ø
72Ø LA$=LA5$(FL) :: GOTO 76Ø
73Ø LA$=LA6$(FL) :: GOTO 76Ø
74Ø LA$=LA7$(FL) :: GOTO 76Ø
75Ø LA$=LA8$(FL)
76Ø DISPLAY AT(12,1):"DER FL
USS FLIESST DURCH:"
77Ø DISPLAY AT(13,3):LA$
78Ø GOTO 38Ø
79Ø ANZ=1
8ØØ IF ST2$(FL)="" THEN 83Ø
ELSE ANZ=ANZ+1
81Ø IF ST3$(FL)="" THEN 83Ø
ELSE ANZ=ANZ+1
82Ø IF ST4$(FL)="" THEN 83Ø
ELSE ANZ=ANZ+1
83Ø IF FG=1 THEN 141Ø
84Ø ON INT(RND*ANZ)+1 GOTO 8
5Ø,86Ø,87Ø,88Ø
85Ø LA$=ST1$(FL):: GOTO 89Ø
86Ø LA$=ST2$(FL):: GOTO 89Ø
87Ø LA$=ST3$(FL):: GOTO 89Ø
88Ø LA$=ST4$(FL)
89Ø DISPLAY AT(12,1):"DER FL
USS FLIESST DURCH:"
9ØØ DISPLAY AT(13,3):LA$
91Ø GOTO 38Ø
92Ø ON INT(RND*2)+1 GOTO 93Ø
,94Ø
93Ø LA$=FL1$(FL):: L$="VON"
:: GOTO 95Ø
94Ø LA$=FL2$(FL):: L$="NACH"
95Ø DISPLAY AT(12,1):"FLIESS
RICHTUNG,"&L$
96Ø DISPLAY AT(13,3):LA$
97Ø GOTO 38Ø
98Ø ANZ=1
99Ø IF NE2$(FL)="" THEN 1Ø2Ø
 ELSE ANZ=ANZ+1
1ØØØ IF NE3$(FL)="" THEN 1Ø2
Ø ELSE ANZ=ANZ+1
1Ø1Ø IF NE4$(FL)="" THEN 1Ø2
Ø ELSE ANZ=ANZ+1
1Ø2Ø IF FG=1 THEN 146Ø
1Ø3Ø ON INT(RND*ANZ)+1 GOTO
1Ø4Ø,1Ø5Ø,1Ø6Ø,1Ø7Ø
1Ø4Ø LA$=NE1$(FL):: GOTO 1Ø8
Ø
1Ø5Ø LA$=NE2$(FL):: GOTO 1Ø8
Ø
1Ø6Ø LA$=NE3$(FL):: GOTO 1Ø8
Ø
1Ø7Ø LA$=NE4$(FL)
1Ø8Ø DISPLAY AT(12,1):"NEBEN
FLUSS:"
1Ø9Ø DISPLAY AT(13,3):LA$
11ØØ GOTO 38Ø
111Ø DISPLAY AT(12,1):"  WIE
 HEISST DER FLUSS ?"
112Ø ACCEPT AT(13,3):F$
113Ø IF F$=FLU$(FL)THEN 158Ø
114Ø DISPLAY AT(12,1):"DAS I
ST FALSCH."
115Ø DISPLAY AT(13,1):"DIE R
ICHTIGE ANTWORT HEISST:"
116Ø DISPLAY AT(14,5):FLU$(F
L)
117Ø FRA=FRA-1
118Ø IF FRA=Ø THEN 166Ø
119Ø FOR I=1 TO 5ØØ :: NEXT
I
12ØØ CALL HCHAR(1Ø,1,32,16Ø)
121Ø TI=Ø :: GOTO 33Ø
122Ø PUNK=PUNK-2 :: IF PUNK<
Ø THEN 151Ø :: DISPLAY AT(1Ø
,1):""
123Ø DISPLAY AT(12,1):"DU WI
LLST ETWAS FRAGEN !"
124Ø ACCEPT AT(13,3):FRAG$
125Ø IF FRAG$="MUENDET" THEN
 52Ø
126Ø IF FRAG$="ENTSPRINGT" T
HEN 55Ø
127Ø IF FRAG$="LAND" THEN 58
Ø
128Ø IF FRAG$="LAENDER" THEN
 FG=1 :: GOTO 58Ø
129Ø IF FRAG$="STADT" THEN 7
9Ø
```

```
1300 IF FRAG$="STAEDTE" THEN
 FG=1 :: GOTO 790
1310 IF FRAG$="FLIESSRICHTUN
G" THEN 920
1320 IF FRAG$="NEBENFLUSS" T
HEN 980
1330 IF FRAG$="NEBENFLUESSE"
 THEN FG=1 :: GOTO 980
1340 DISPLAY AT(12,1):"ICH W
EISS NICHT, WAS DU" :: DISPL
AY AT(13,3):"VON MIR WILLST.
"
1350 FOR I=1 TO 500 :: NEXT
I :: GOTO 1230
1360 IF ANZ>1 THEN 1390
1370 DISPLAY AT(14,1):"ER FL
IESST DURCH EIN LAND"
1380 GOTO 380
1390 DISPLAY AT(14,1):"ER FL
IESST DURCH "&STR$(ANZ)&" LA
ENDER"
1400 GOTO 380
1410 IF ANZ>1 THEN 1440
1420 DISPLAY AT(14,1):"ER FL
IESST DURCH EINE STADT"
1430 GOTO 380
1440 DISPLAY AT(14,1):"ER FL
IESST DURCH "&STR$(ANZ)&" ST
AEDTE"
1450 GOTO 380
1460 IF ANZ>1 THEN 1490
1470 DISPLAY AT(14,1):"ICH K
ENNE EINEN NEBENFLUSS"
1480 GOTO 380
1490 DISPLAY AT(14,1):"ICH K
ENNE "&STR$(ANZ)&" NEBENFLUE
SSE"
1500 GOTO 380
1510 DISPLAY AT(12,1):"DU BR
AUCHST ZU LANGE."
1520 CALL HCHAR(13,1,32,64)
1530 TI=0 :: FRA=FRA-1
1540 IF FRA=0 THEN 1660
1550 FOR I=1 TO 500 :: NEXT
I
1560 CALL HCHAR(10,1,32,128)
1570 GOTO 1150
1580 DISPLAY AT(12,1):"RICHT
IG"
1590 DISPLAY AT(13,3): "PUNKT
E :"&STR$(PUNK)
1600 PUN=PUN+PUNK
1610 TI=0
1620 FRA=FRA-1 :: IF FRA=0 T
HEN 1660
1630 FOR I=1 TO 500 :: NEXT
I
1640 CALL HCHAR(10,1,32,128)
1650 GOTO 330
1660 PRINT
```

LISTINGS

```
1670 PRINT "SPIEL ENDE"
1680 PRINT "PUNKTE :"&STR$(P
UN)
1690 END
1700 CALL CLEAR :: DISPLAY A
T(12,1):"ES GEHT GLEICH LOS,
ICH MUSS" :: DISPLAY AT(13,1
):"NUR NOCH EINIGE DATEN LES
EN."
1710 RESTORE 1720
1720 DATA CNM@T,SCHWARZE MEE
R,SCHWARZWALD,DEUTSCHLAND,OE
STERREICH,CSSR,UNGARN,JUGOSL
AWIEN,RUMAENIEN
1730 DATA BULGARIEN,RUSSLAND
,WIEN,BUDAPEST,,,WESTEN,OSTE
N,INN,DRAU,MORAVA,THEISS
1740 DATA DKAD,NORDSEE,RIESE
NGEBIRGE,CSSR,DDR,DEUTSCHLAN
D,,,,,,DRESDEN,HAMBURG,,,SUE
D-OST,NORD-WEST,SAALE,MOLDAU
,HAVEL,EGER
1750 DATA FT@C@KPTHUHQ,ATLAN
TIK,SERRIAS BETICAS,SPANIEN,
,,,,,,SEVILLA,,,,OSTEN,WEST
EN,GENIL,GUADIANA MENOR,,
1760 DATA FT@CH@M@,ATLANTIK,
SIERRA MORENCE,SPANIEN,PORTU
GAL,,,,,ICH KENNE KEINE GR
OSSE STADT,,,,OSTEN,WESTEN,Z
UJAR,ZANCARA,,
1770 DATA DAQN,MITTELMEER,KA
NTABRISCHES GEBIRGE,SPANIEN,
,,,,,,ZARAGOZA,,,,NORDEN,SU
EDEN,JALON,GALLEGO,SEGRE,ARA
GON
1780 DATA F@QNMMD,GOLF VON B
ISCAYA,PYRENAEEN,SPANIEN,FRA
NKREICH,,,,,,,TOULOUSE,BORDE
AUX,,,SUEDEN,NORDEN,LOT,AVEY
RON,GERS,DORDOGNE
1790 DATA KNHQD,GOLF VON BIS
CAYA,FRANZ. MITTELGEBIRGE,FR
ANKREICH,,,,,,NANTES,,,SU
ED-OST,NORD-WEST,ALLIER,CHER
,VIENNE,INDRE
1800 DATA RDHMD,AERMELKANAL,
FRANZ. MITTELGEBIRGE,FRANKRE
ICH,,,,,,,PARIS,,,,SUED-OST
,NORD-WEST,AUBE,MARNE,OISE,E
URE
1810 DATA QGNMD,MITTELMEER,A
LPEN,FRANKREICH,SCHWEIZ,,,
,,GENF,LYON,,,NORD-OST,SUED-
WEST,DURANCE,DROME,ISERE,GAR
D
1820 DATA L@@R,NORDSEE,FRANZ
. MITTELGEBIRGE,FRANKREICH,B
ELGIEN,NIEDERLANDE,,,,,,LUET
TICH,,,,SUEDEN,NORDEN,SAMBRE
,WAAL,,
```

LISTINGS

```
1830 DATA QGDHM, NORDSEE, ALPE
N, SCHWEIZ, FRANKREICH, DEUTSCH
LAND,NIEDERLANDE, ,,,,BASEL,D
UESSELDORF ,MAINZ, , SUEDEN, NOR
DEN
1840 DATA MAIN,MOSEL, NECKAR,
SIEG
1850 DATA CNTQN,ATLANTIK, IBE
RISCHES RANDGEBIRGE, SPANIEN,
PORTUGAL,,,,,,,PORTO,,,,OSTE
N, WESTEN, TORMES,ESLA, TUA, HUE
BRA
1860 DATA VDRDQ,NORDSEE, THUE
RINGER WALD, DEUTSCHLAND, DDR,
,,,,,,BREMEN,,,,SUEDEN, NORDE
N, ALLER, FULDA, WERRA,
1870 DATA LNRDK, RHEIN, VOGESE
N, DEUTSCHLAND, FRANKREICH, LUX
EMBURG,,,,,,KOBLENZ,,,,NORDE
N, SUEDEN, SAAR, MEURTHE,,
1880 DATA HMM, DONAU, ALPEN, OE
STERREICH, SCHWEIZ, DEUTSCHLAN
D,,,,,,INSBRUCK,,,,NORD-OST,
SUED-WEST,SALZACH,,,
1890 DATA ON,ADRIATISCHES ME
ER, ALPEN, ITALIEN,,,,,,,,TURI
N, ,,,WESTEN, OSTEN, ADDU, OGLIO
,TREBBIA, TIEINO
1900 DATA SHADQ, MITTELMEER, A
PENNINEN, ITALIEN,,,,,,,,ROM,
, ,,NORDEN, SUEDEN,NERA,,,
1910 DATA NCDQ,OSTSEE, SUDETE
N,POLEN,CSSR,,,,,,,BRESLAU,S
TETTIN, , , SUEDEN, NORDEN, BOBER
,WARTHE,,
1920 DATA VDHBGRDK, OSTSEE,KA
RPATEN, POLEN, , , , , , , , WARSCHAU
,KRAKAU, , , SUEDEN, NORDEN, SAN,
PILIEN,NOGAT, WIEPRZ
1930 DATA SGDHRR, DONAU,KARPA
TEN, RUSSLAND, UNGARN, JUGOSLAW
IEN,,,,,,ICH KENNE KEINE GRO
SSE STADT,,,,NORDEN, SUEDEN,S
ZAMOS,MURES, BODROA,
1940 DATA LNQ@U@, DONAU, DINAR
ISCHES GEBIRGE, JUGOSLAWIEN,,
,,,,,,ICH KENNE KEINE GROSSE
 STADT, ,,,SUEDEN, NORDEN, NISA
VA,WESTL. MORAVA,,
1950 DATA RDUDQM, ATLANTIK, CA
MBRION MOUNTAINS,ENGLAND, WAL
ES,,,,,,,ICH KENNE KEINE GRO
SSE STADT, ,,,NORDEN, SUEDEN, A
VON,,,
1960 DATA SGDLRD,NORDSEE, COT
SWOLD HILLS,ENGLAND,,,,,,,L
ONDON, ,,,WESTEN, OSTEN, ICH KE
NNE KEINEN GROSSEN,,,
1970 DATA R@UD,DONAU, ALPEN,J
UGOSLAWIEN,,,, , ,,,BELGRAD,ZA
GREB, ,,WESTEN, OSTEN, UNA, KUPA
, BORSNA, DRINA
1980 DATA CQ@T, DONAU, ALPEN, I
TALIEN, OESTERREICH, JUGOSLAWI
EN, ,,,,,MARBURG,OSIJEK,,,WES
TEN,OSTEN,MUR,,
1990 DATA VNKF@,KASPISCHES M
EER,MITTELRUSS. PLATTE, RUSSL
AND,,, , , , , , JUROSLAWL, WOLGAGR
AD,SARATOW,KUIBYSCHEW,NORD-W
EST, SUED-OST,KAMA,OKA,,
2000 FOR I=1 TO 26
2010 READ FLU$(I),MU$(I),EN$
(I),LA1$(I),LA2$(I),LA3$(I),
LA4$(I),LA5$(I),LA6$(I),LA7$
(I),LA8$(I),ST1$(I),ST2$(I),
ST3$(I),ST4$(I)
2020 READ FL1$(I),FL2$(I),NE
1$(I),NE2$(I),NE3$(I),NE4$(I
)
2030 NEXT I
2040 FOR I=1 TO 26 :: T$=""
2050 FOR U=1 TO LEN(FLU$(I))
2060 C$=SEG$(FLU$(I),U,1)
2070 C$=CHR$(ASC(C$)+1)
2080 T$=T$&C$
2090 NEXT U
2100 FLU$(I)=T$
2110 NEXT I
2120 RETURN
2130 DISPLAY AT(6,5)SIZE(3):
USING "###":PUN :: DISPLAY A
T(6,17)SIZE(2):USING "##":FR
G
2140 RETURN
```

TI REVUE

Das Magazin für TI 99-4A

Nr. 8/87

Mit Bauplan: 8-Kanal Analog / Digital-Wandler

Gewusst Wie: Der Disc-Controller

Ent-schleiert: Mysterium Kasette

DAS HEFT IM HEFT FÜR TI-USER

EINFACH HERAUSTRENNEN

DISC-CONTROLLER UND DISKETTEN BEIM TI 99 4/A

Nachdem es für den TI99/4A schon eine Menge Programme bzw. Module gibt, die es ermöglichen, sich auf der Diskette die Daten direkt anzuschauen und zu ändern – die sogenannten Disk-Fixer – fand ich es als eine Notwendigkeit, mal all die Grundkenntnisse, die dafür benötigt werden, zusammenzufassen.
Es ist aber etwas mehr daraus geworden als ich mir ursprünglich vorgenommen hatte, weil eine sinnvolle Anwendung dieser Programme nur möglich wird, wenn man die Bedeutung jedes einzelnen Bit – mindestens die der wichtigsten Diskettensektoren – kennt. Auf diese Weise hat man z.B die Möglichkeit, versehentlich gelöschte Dateien wieder herzustellen, Disketten, deren Katalog nicht mehr aufzulisten ist, wieder zu „reparieren", Disketten- und Datenschutz zu ändern, um z.B. eigene Disketten zu schützen, bis hin zur Wiederherstellung von Dateien, wenn mal der Diskette versehentlich etwas passiert ist Außerdem ist das Grundprinzip der Datenaufzeichnung, wie auch die Einteilung der Diskette, nicht nur für den TI99/4A gültig – die erworbenen Kenntnisse kann man auch anderweitig anwenden.

1. ALLGEMEINES

Der Disk-Controller ist die Steuereinheit, die die Ein- und Ausgabe von Daten zwecks Speicherung auf Diskette verwaltet. Der Disk-Controller von Texas Instruments kann bis zu 3 Laufwerke mit 5.25-Zoll-Disketten steuern. Die Speicherung der Daten erfolgt in einfacher Dichte (SD=single density), entweder einseitig (SS=single sided) oder doppelseitig (DS=double sided). Pro Seite werden 40 Spuren (tracks 0–39) gebildet, wobei jede Spur aus 9 Sektoren (0–8) besteht. Ein Sektor enthält 256 Bytes und kann als einheitliches Datenbündel betrachtet werden, weil dies die kleinste Einheit von Daten ist, die gemeinsam auf die Diskette geschrieben, bzw. von der Diskette gelesen werden kann. Insgesamt können pro Seite 40 x 256 = 92160 Bytes, d.h., genau 90 kByte gespeichert werden. Mit einem doppelseitigen Laufwerk, mit zwei Schreib-Leseköpfen, können es bis 180 kByte sein.
Zum Unterschied zu dem TI-Controller kann der Disk-Controller von CorComp bis zu 4 Laufwerke steuern, wie auch Daten mit doppelter Dichte, und zwischen 18 Sektoren (0–17) pro Spur speichern, was eine Gesamtkapazität von 360 kByte ergibt. Auch bei diesem Controller ist ein Sektor 256 Byte lang.
Die Spur 0 befindet sich am äußeren Rand der Diskette und die Spur 39 am inneren Rand; dies gilt auch bei doppelseitigen Disketten und zwar für beide Seiten. Was aber nun die Sektoren betrifft, ist deren Anordnung bei doppelseitigen Disketten etwas anders. Die Sektoren 0 bis 359 befinden sich auf der ersten Seite von außen nach innen und die Sektoren 360 bis 719 auf der zweiten Seite von innen nach außen. Das folgende Bild soll dies verdeutlichen:

a) Anordnung der Spuren (tracks) bei doppelseitigen Disketten:

```
(Seite 2)          39------<------0
----------- Mitte -----------------
(Seite 1)          39------<------0
```

b) Anordnung der Sektoren bei doppelseitigen Disketten:

```
(Seite 2)          360------>------719
----------- Mitte --------------------
(Seite 1)          359------<------0
```

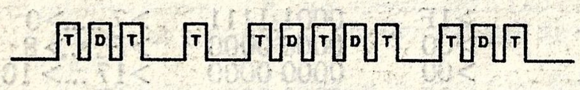
Zur Formatierung der Diskette wird die sogenannte Soft-Sektorierungsmethode verwendet, wobei die Einteilung der Sektoren pro Spur softwaremäßig erfolgt. Die Diskette hat im Gegensatz zur Hard-Sektierung nur ein Indexloch, um die Positionierung der Spuren zu lokalisieren.
Wie aus den folgenden Tabellen 1 und 2 hervorgeht, wird die Position der Sektoren in dem ID- oder Kennungs-Adressfeld abgelegt. Darin sind die Spurnummer (0–8, bzw. 0–17), die Seitennummer (1 oder 2), die Sektornummer (0 bis 8), die Sektorlänge (256 Bytes) und die CRC-Fehlerbytes enthalten. Nach diesem ID-Adressfeld folgt nach einigen Trennungsbytes das Datenfeld mit den eigentlichen Sektordaten.
Die Lücke mit den Trennungsbytes hat zwei wichtige Funktionen: Erstens erfolgt auf diesem Abschnitt die Anpassung der Motorgeschwindigkeit des Schrittmotors der Schreib-Leseköpfe und zweitens der vollständige Abbau bzw. Aufbau des Magnetfeldes in den Schreib-Leseköpfen, während diese vom Lesen der Daten aus dem ID-Adressfeld zum Beschreiben der Sektoren im Datenfeld hin- und hergeführt werden.
Der Disk-Controller hat, außer der Steuerung der Schreib-Leseköpfe und dem Auffinden der Spuren und Sektoren noch die Aufgabe, die parallel ankommenden Daten vom Rechner in seriellen Daten zwecks Abspeicherung auf Diskette umzuwandeln und weiterzugeben – natürlich auch umgekehrt, beim Lesen der Daten. Es werden dabei je 8 Bit zu einem Byte zusammengefaßt, d.h., beim Lesen eines Sektors muß der Controller 2048 Bits zu 256 Bytes zusammenfassen und weitergeben. Gleichfalls die zusätzlichen Bits für die Synchronisation (ID-SYNC und DATEN-SYNC), die Adressbytes, anhand derer die Spuren und Sektoren lokalisiert werden können, sowie die CRC-Bytes (Cyclic Redundancy Check), die der Fehlererkennung dienen. Beim Abspeichern ist der Vorgang genau umgekehrt, wobei aber nach Abspeichern jedes einzelnen Sektors dieser nochmals gelesen und mit dem aktuellen Pufferinhalt verglichen wird. Im Mißerfolgsfall, beim Lesen oder Schreiben der Daten, wird der Vorgang noch bei den nächsten 5 Umdrehungen der Diskette, d.h., höchstens 5mal, wiederholt. Ist auch dies erfolglos, so sendet der Controller dem Betriebssystem einen Fehlercode, woraufhin eine Fehlermeldung ausgegeben wird.

2. AUFZEICHNUNG DER DATEN

a) Die Aufzeichnung der Daten mit einfacher Dichte (SD) wird mit der Frequenzmodulations-Methode (FM) realisiert. Jede Informationseinheit (Bit-Zelle) enthält einen Takt-Puls (T) und zusätzlich, nur wenn das betreffende Bit den Wert 1 hat, auch einen Daten-Puls (D), wie das folgende Beispiel zeigt:

|bit 7|bit 6|bit 5|bit 4|bit 3|bit 2|bit 1|bit 0|

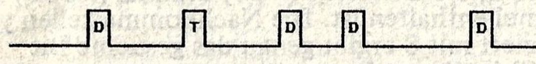
Bitwert: 1 0 0 1 1 0 1 0 = >9A



b) Die Aufzeichnung der Daten mit doppelter Dichte (DD) wird mit der sogenannten Modifizierten Frequenzmodulations-Methode (MFM) realisiert. Die Informationseinheit erhält nur dann einen Daten-Puls (D), wenn das betreffende Bit den Wert 1 hat. Enthalten zwei aufeinanderfolgende Informationseinheiten den Bitwert Null, also keinen Daten-Puls (D), dann wird ein Takt-Puls (T) generiert, wie das folgende Beispiel zeigt:

|bit 7|bit 6|bit 5|bit 4|bit 3|bit 2|bit 1|bit 0|

Bitwert: 1 0 0 1 1 0 1 0 = >9A



3. AUFBAU DER DISKETTEN-SPUR

a. Das Format der Diskette mit einfacher Dichte (SD)

Bei der Initialisierung einer Diskette werden folgende Informationen aufgezeichnet:

Nr	Bezeichnung	Bytes	Inhalt	Bedeutung
1	Vorspann	12	> FF	Vorspann-Schutzbytes
2	ID-SYNC-Bytes	6	> 00	Synchronisationsbytes für den Controller (Bit-Level-Synchronisation)
3	ID-Adress-Marke	1	> FE	Adressfeld-Identifizierung (Start des ID-Feldes)
4	TRACK-Adresse	1	Spur-Nummer	- von >00 bis >27 für die Tracks 0 bis 39
5	DISK-Seite	1	Seiten-Nummer	>00 für Seite 1 und >01 für Seite 2
6	SEKTOR-Adresse	1	Sektor-Nummer	- von >00 bis >08 für die Sektoren 0 bis 8 die sich auf dieser Spur befinden
7	SEKTOR-Länge	1	> 01	- d.h. 256 Bytes / Sektor
8	ID-CRC-Bytes	2	CRC-Wert	- zur Fehlerüberprüfung der ID-Bytes
9	ID-Trennungs-Bytes	11	> FF	- Lücke zwischen ID- und Datenfeld (siehe Text)
10	DATEN-SYNC-Bytes	6	> 00	- zur Synchronisation für die Informationseinheiten
11	DATEN-Adressmarke	1	> FB	Datenfeld-Identifizierung
12	D A T E N	256	Sektor-Daten	- eigentliche Daten des betreffenden Sektors
13	DATEN-CRC-Bytes	2	CRC-Wert	- zur Fehlerüberprüfung der Datenbytes
14	DATEN-Trennungs-Bytes	36	> FF	- Lücke zwischen diesem Datenfeld und dem ID-Feld des nächsten Sektors
15	Nachspann	240	> FF	Nachspann-Schutzbytes

- T A B E L L E 1 -

b. Das Format der Diskette mit doppelter Dichte (DD)

Nr	Bezeichnung	Bytes	Inhalt	Bedeutung
1	Vorspann	32	> 4E	Vorspann-Schutzbytes
2	ID-SYNC-Bytes	12 3	> 00 > A1	Synchronisationsbytes für den Controller (Bit-Level) - diese Bytes sind nur MCM-spezifisch (-> Text)
3	ID-Adress-Marke	1	> FE	Adressfeld-Identifizierung (Start des ID-Feldes)
4	TRACK-Adresse	1	Spur-Nummer	- von >00 bis >27 für die Tracks 0 bis 39
5	DISK-Seite	1	Seiten-Nummer	>00 für Seite 1 und >01 für Seite 2
6	SEKTOR-Adresse	1	Sektor-Nummer	- von >00 bis >08 für die Sektoren 0 bis 8 die sich auf dieser Spur befinden
7	SEKTOR-Länge	1	> 01	- d.h. 256 Bytes / Sektor
8	ID-CRC-Bytes	2	CRC-Wert	- zur Fehlerüberprüfung der ID-Bytes
9	ID-Trennungs-Bytes	22	> 4E	- Lücke zwischen ID- und Datenfeld (siehe Text)
10	DATEN-SYNC-Bytes	12 3	> 00 > A1	- zur Synchronisation für die Informationseinheiten - diese Bytes sind nur MCM-spezifisch (-> Text)
11	DATEN-Adressmarke	1	> FB	Datenfeld-Identifizierung
12	D A T E N	256	Sektor-Daten	- eigentliche Daten des betreffenden Sektors
13	DATEN-CRC-Bytes	2	CRC-Wert	- zur Fehlerüberprüfung der Datenbytes
14	DATEN-Trennungs-Bytes	28	> 4E	- Lücke zwischen diesem Datenfeld und dem ID-Feld des nächsten Sektors
15	Nachspann	190	> 4E	Nachspann-Schutzbytes

- T A B E L L E 2 -

Bemerkungen zur Tabelle 1:

Der Bereich 2–9 ist das ID- oder Kennungsfeld und der Bereich 10–14 das Datenfeld.

Der gesamte Bereich 2–14, d.h., zwischen Vorspann und Nachspann, ist 9mal auf jeder Spur enthalten – entsprechend den 9 Sektoren. Dies sind insgesamt 325 x 9 = 2925 Bytes. Zusammen mit Vor- und Nachspann sind also 2925 x 12 + 240 = 3177 Bytes auf einer Spur. Davon sind aber nur 256 x 9 = 2304 reine Sektordaten-Bytes.

Bemerkungen zur Tabelle 2:

Der Bereich 2–14, d.h., zwischen Vorspann und Nachspann, ist 18mal auf jeder Spur enthalten – entsprechend den 18 Sektoren. Dies sind insgesamt 346 x 18 = 6228 Bytes. Zusammen mit Vor- und Nachspann sind also 6228 + 32 + 190 = 6450 Bytes auf einer Spur. Davon sind aber nur 256 x 18 = 4608 reine Sektordaten-Bytes.

Der Unterschied zwischen den beiden Formaten (SD und DD) besteht erstens darin, daß zusätzlich zu den Synchronisationsbytes (ID-SYNC und DATEN-SYNC) noch je 3 Bytes (>A1) benötigt werden, was nur MFM-spezifisch ist – und zweitens ist die gesamte Speicherkapazität genau doppelt so groß.

4. AUFBAU DER DISKETTEN-SEKTOREN

Was die Sektoren der Disketten betrifft, so kann man

diese nach dem Informationsinhalt in 4 große Gruppen einteilen:
A. Der Sektor 0 – enthält die allgemeinen Informationen zur Diskette, angefangen mit dem Diskettennamen bis zu der Belegungsinformation, der sogenannten Bit-Map (siehe Tabelle 3).
B. Der Sektor 1 – enthält die Link-Map; das ist die Verzweigungstabelle, die erstens zur Katalogbildung dient und zweitens zum Auffinden der Programme auf der Diskette. Hier sind die belegten Sektoren aufgelistet und zw. in der Reihenfolge, wie sich die Dateinamen, in alphabetischer Reihenfolge auf der Diskette befinden. Das scheint zwar etwas verwirrt, doch werde ich später genauer darauf eingehen (siehe auch Tabelle 4).
C. Die Sektoren >2 bis >21 – enthalten alle Namen und Informationen der einzelnen Dateien wie Dateityp, Anzahl und Lokalisierungsdaten der von jeder Datei belegten Sektoren. Jeder Datei, die auf Diskette gespeichert wird, wird ein Sektor aus diesem Bereich zugeordnet (siehe Dahbelle 5).
D. Die Sektoren >22 bis >168 (bzw. >2D0 oder >5A0) – enthalten die eigentlichen Daten oder einzelnen Dateien.
Um einen besseren Überblick zu gewährleisten, werde ich nun diese Gruppen in Tabellenform zusammenfassen und beschreiben, praktisch auf die Bedeutung jedes einzelnen Bit eingehen und dessen Funktion kurz erläutern. Nachfolgend sind Hexadezimalwerte mit einem davorstehenden '>'-Zeichen gekennzeichnet. Fehlt dieses Zeichen, so handelt es sich immer um Dezimalwerte.

A. DER SEKTOR 0

TABELLE 3:

BYTE (hex)	Bedeutung
>0 - >9	Diskettenname; alle Zeichen erlaubt außer Leerzeichen
>A - >B	Anzahl der formatierten Sektoren auf der Diskette: >168 (=360 Sektoren) bei SS/SD-Laufwerken >2D0 (=720 Sektoren) bei SS/DD und DS/SD-Laufwerken >5A0 (=1440 Sektoren) bei DS/DD-Laufwerken
>C	Anzahl der Sektoren auf jeder Spur: >09 (= 9 Sektoren) bei einer Aufzeichnung mit SD >12 (= 18 Sektoren) bei einer Aufzeichnung mit DD
>13->15	In diesen Bytes steht immer 'DSK' - dies ist das Erkennungszeichen daß die Diskette mit einem TI- oder kompatiblen Controller initialisiert wurde
>10	Diskettenschutz: >20 wenn die Diskette nicht geschützt ist (normal) >50 (="P" wie Protected),wenn sie geschützt ist
>11	Anzahl der Spuren pro Seite: >23 = 35 Spuren pro Seite >28 = 40 Spuren pro Seite
>12	Anzahl der Seiten: >01 = einseitig (SS) >02 = doppelseitig (DS)
>13	Disketten-Dichte: >01 = einfache Dichte (SD) >02 = doppelte Dichte (DD)
>14->37	- nicht benutzt; ausgefüllt mit Nullen
>38->FF	Belegungsinformationen der Diskette (Bit Map). Jedem Sektor entspricht ein Bit - nur wenn dieses 1 ist, dann ist der betreffende Sektor belegt. Diese 200 Bytes reichen aus um 1600 Sektoren zu verwalten, bzw. 400 kByte; unsere Disketten beinhalten aber nur maximal 360 kByte (wenn DS/DD). Jedem Byte aus diesem Bereich entsprechen somit 8 Sektoren auf der Diskette. Dieser Bereich wird wie folgt benutzt: >38->64 (=45 Bytes x 8 Bit) = 360 Sektoren bei SS/SD >38->91 (=90 Bytes x 8 Bit) = 720 Sektoren bei SS/DD oder bei DS/SD >38->EB (180 Bytes x 8 Bit) =1440 Sektoren bei DS/DD Die restlichen Bytes bis >FF haben alle den Wert FF.

Beispiel:

Bytes:	Wert (Beispiel)	Binär	Zugeordn. Sektoren
>38	>1F	0001 1111	>7 ....>0
>39	>00	0000 0000	>F ....>8
>3A	>00	0000 0000	>17 ...>10
>3B	>00	0000 0000	>1F ...>18
>3C	>7C	0111 1100	>27 ....>20

Demnach sind die Sektoren >0, >1, >2, >3, >4 sowie >22, >23, >24, >25, >26 der Diskette belegt.

Zwei wichtige Berechnungsformeln:
a) Welches Bit ist zuständig für einen bestimmten Sektor und in welchem Byte der Bit-Map ist dieses Bit?

(Sektornummer, dezimal : 8) + 56 = xxx,yyy

Der ganzzahlige Anteil xxx ist das gesuchte Byte und zwar in der dezimalen Schreibweise, da auch das Anfangsbyte dieses Bereiches (>38) dezimal (als 56) in der Formel enthalten ist. Die Nachkommastellen yyy multipliziert mit 8 ergibt genau das gesuchte Bit (>0 >7) in diesem Byte.

Beispiel:
Welches Bit ist zuständig für den Sektor >35 (= dezimal 53) und in welchem Byte der Bitmap befindet es sich?
Berechnung: (53 : 8) + 56 = 62,625
Das gesuchte Bit befindet sich im Byte 62, d.h., >3E in der Bit-Map des Sektors 0 und das zuständige Bit ist 0,625 x 8 = 5, also Bit 5:
Bitposition: 7 6 6 4 3 2 1 0
Bitwert (z.B.):

Bitposition: 7 6 [5] 4 3 2 1 0
Bitwert (z.B.): 0 1 [1] 1 1 1 1 1 = >7F
Ist der Wert dieses Bytes, wie hier beispielsweise angenommen >7F (binär: 0111 1111), dann ist der Sektor >35 belegt, da das Bit 5 den Wert 1 hat.
b) Welche Sektoren (immer 8!) werden von einem bestimmten Byte der Bit-Map verwaltet?

(Bytenummer,dezimal - 56) x 8 = zzz

Der Wert zzz ist der erste Sektor der gesuchten 8 Sektoren-Gruppe.

Beispiel:
Welche Sektoren werden von dem Byte >5E (dezimal 94) der Bit-Map verwaltet?
Berechnung: (94 – 56) x 8 = 304 also hex. >130 d.h., die gesuchten 8 Sektoren sind >130, >131,... >137.

B. DER SEKTOR 1

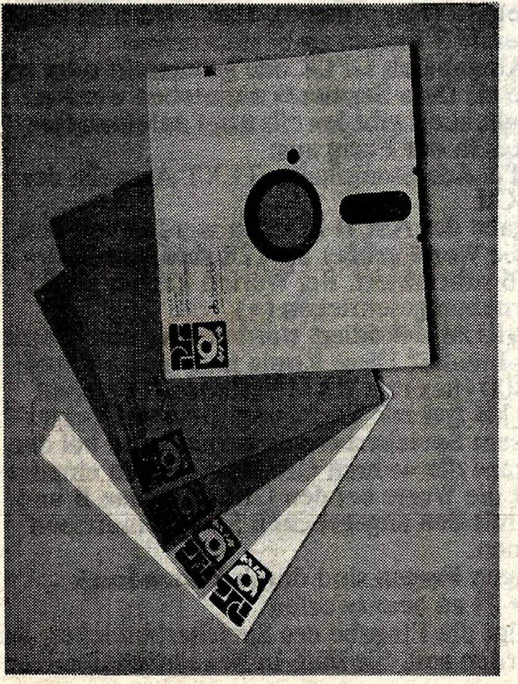
BYTE (hex)	Bedeutung
>0 - >1	Pointer für die alphabetisch 1-te Datei auf der Diskette - siehe Bemerkung *).
>2 - >3	Pointer für die alphabetisch 2-te Datei auf der Diskette.
>4 - >5	Pointer für die alphabetisch 3-te Datei auf der Diskette.
u.s.w. bis FF	- maximal 128 Pointer mit einer Länge von je 2 Bytes

– TABELLE 4 –

***)Bemerkung:**
Unter Pointer ist hier ein Zeiger auf denjenigen Sektor aus dem Bereich >02 – >21 gemeint, in dem sich die Dateiinformationen, d.h., die spezifischen Daten jeder einzelnen Datei befinden. Ein Beispiel soll dies verdeutlichen:
Auf eine Diskette werden 4 Dateien in folgender Reihenfolge gespeichert: A-DAT, C-DAT, D-DAT und B-DAT. Ihnen werden gleich nach dem Abspeichern die Sektoren >0002, >0003, >0004 und >0005 als Dateiinformations-Sektoren zugeordnet, also:
A-DAT hat nun den Dateiinformations-Sektor >0002
C-DAT hat nun den Dateiinformations-Sektor >0003
D-DAT hat nun den Dateiinformations-Sektor >0004
B-DAT hat nun den Dateiinformations-Sektor >0005
Wie man sieht, sind diese Dateien nicht in alphabetischer Reihenfolge gespeichert worden. Betrachtet man nun diese 4 Dateien in alphabetischer Reihenfolge mit ihren dazugehörigen Dateiinformations-Sektoren, so ergibt sich folgende Aufstellung:
A-DAT, mit dem Dateiinformations-Sektor >0002
B-DAT, mit dem Dateiinformations-Sektor >0005
C-DAT, mit dem Dateiinformations-Sektor >0003
D-DAT, mit dem Dateiinformations-Sektor >0004
Genau das sind nun die Werte die in den Sektor 1 eingetragen werden, und zwar:

Byte:	Inhalt:
>0– 1	0002 = Pointer auf d. alphab. 1te Datei
>2– 3	0005 = Pointer auf d. alphab. 2te Datei
>4– 5	0003 = Pointer auf d. alphab. 3te Datei
>6– 7	0004 = Pointer auf d. alphab. 4te Datei
>8– 9	0000
....	......
bis >FF	0000

Dieser Sektor kann maximal 128 solcher Pointer aufnehmen, was der gleichen Anzahl von Dateien entspricht, die auf die Diskette gespeichert werden können.



C. DIE SEKTOREN 2 – 21

BYTE	Bedeutung
>0 – >9	Dateiname; alle Zeichen erlaubt außer Leerzeichen
>A – >B	nicht benutzt; ausgefüllt mit Nullen
>C	Dieses Byte gibt Aufschluß über die Art der Datei: >00 = DIS/FIX >80 = DIS/VAR >01 = PROGRAM >82 = INT/VAR >02 = INT/FIX Zu diesem Wert wird noch >08 dazugezählt wenn die Datei schreibgeschützt ist. Bedeutung der einzelnen Bits: Bit: / Wert=1 / Wert=0 >0 PROGRAM-file / DATA-file >1 INTERNAL-Format(binär) / DISPLAY-Format(ASCII) >2 - nicht verwendet / - nicht verwendet >3 Schreibschutz / kein Schreibschutz >4,5,6 - nicht verwendet / - nicht verwendet >7 Datensätze mit VARIABLER Länge / Datensätze mit FIXER Länge
>D	Dieses Byte ist nur für DATA-files mit FIXER Länge relevant. Hier bedeutet es die maximale Anzahl der Datensätze pro Sektor. Der Wert dieses Bytes ergibt sich aus der Sektorlänge (256) geteilt durch die Länge der Datensätze; z.B. für eine DIS/FIX 80-Datei: 256 : 80, bzw. >FF : >50 = >03. - Bei PROGRAM-files ist der Wert immer >00 und bei Dateien mit Datensätzen VARIABLER Länge immer >01
>E – >F	Anzahl der von der Datei belegten Sektoren, ab dem Sektor >22; diese ist immer um 1 kleiner als es der Disk-Manager für die betreffende Datei anzeigen würde, weil dieser auch den Sektor mit den Datei-Informationen (>02 – >21) mitzählt.
>10	Dieses Byte ist nur für PROGRAM-files und DATA-files mit VARIABLER Länge relevant. Hier bedeutet es die Anzahl der Rest-Bytes im letzten, von dieser Datei belegten Sektor. Als Abschluß folgt dann noch eine Datei-Endemarkierung die bei PROGRAM-files >AA und bei DATA-files >FF ist. - Bei Dateien mit Datensätzen FIXER Länge hat dieses Byte (>10) den Wert >00.
>11	Dieses Byte ist nur für DATA-files relevant. Es stellt die logische Länge der Datensätze dar, d.h. die maximale Anzahl der Bytes die pro Datensatz eingelesen werden sollen. Zum Beispiel bei FIX- oder VAR-80- Dateien sind dies 80 Bytes.
>12–>13	Diese Bytes sind nur für DATA-files relevant, u.zwar: a. bei DATA-files mit VARIABLER Länge bedeuten sie die Anzahl der, von der Datei belegten Sektoren; dieser Wert ist identisch mit dem der Bytes >E–>F. Diese 2 Bytes müssen vertauscht interpretiert werden, d.h. zuerst das niederwertige und dann das höherwertige Byte – zum Beispiel: >1A00 wird als >001A interpretiert b. bei DATA-files mit FIXER Länge bedeuten sie die Anzahl der Datensätze die die Datei insgesamt belegt. Diese 2 Bytes werden gleichfalls vertauscht interpretiert.
>14–>1B	- nicht benutzt; ausgefüllt mit Nullen
>1C–>FF	Dieser Bereich beinhaltet den sogenannten Daten - Pointer-Block. Je 3 Bytes bilden eine Aussage über Blockverknüpfungen von Datensegmenten – siehe Erklärung **) weiter unten.

****)Erklärung:**
Wenn sich zum Beispiel eine Datei, während sie weiterentwickelt wird, entsprechend vergrößert, so reicht beim Zurückspeichern auf die Diskette, der vorher belegte Speicherplatz nicht mehr aus.
Deshalb muß am Ende der bereits belegten Sektoren ein neuer Sektor mit dem Rest der Datei A-DAT angelegt werden. Nun werden alle Adressen der Sektoren, die noch derartige „Dateisplitter" enthalten, in diesem Sektorbereich >1C – >FF notiert, damit sie der Controller beim Einlesen wieder findet. Beim Umkopieren einer Diskette (nur dateiweise!), wird dann wieder Ordnung hergestellt und jede Datei belegt nur noch aufeinanderfolgende Sektoren. Dies wirkt sich natürlich dann beim Laden der Datei auch auf die Geschwindigkeit aus, mit der die Daten eingelesen

werden – die Dateien sind dann wieder „geordnet". Nun zurück zu den Bytes dieses Sektors. Es werden also 3-Byte-Gruppen gebildet, worin die notwendigen Informationen über solche Blockverknüpfungen von Dateisegmenten eingetragen werden. Anhand eines Beispiels will ich dies mal kurz erklären:

Byte:	>1C	>1D	>1E	>1F	>20	>21
Inhalt (z.B.):	>21	>61	>01	>31	>41	>01
Vertauscht:	>01	>61	>21	>01	>41	>31

d.h.,der erste Block der Datei befindet sich im Sektor >121
d.h., die Anzahl der belegten Sektoren, ab dem Sektor >121 ist >16 (dez. 22).

Dies ist aber erst die Gesamtzahl der Sektoren, die die Datei bis *hierher* belegt. Der nachfolgende 3er-Block wird dann auch (dezimal 20) Sektoren ab >131 belegt. Folgen noch solche 3er-Gruppen, deren Wert verschieden von Null ist, dann würde die Datei noch weitere Sektoren auf der Diskette belegen.

DOLMETSCHER

Das Programm stellt einen in der Programmiersprache „C" geschriebenen GPL-Disassembler für den Heimcomputer TI99/4A dar. Alle Programme wurden mit dem e/A-Modulpaket und dem C-Compiler von C. Pulley Version 2.0 erstellt.

Konfiguration:
Benötigt werden neben der Konsole:
- Editor/Assembler-Paket
- C-Compiler v. C. Pulley 2.0
- 32k Ram-Erweiterung
- mindestens 1 Diskettenlaufwerk

Programmbeschreibung:
Das Programm ist in der Programmiersprache „C" verfaßt. Es übersetzt die Tokens der Sprache GPL in vom Benutzer zu verstehende Befehlsfolgen. Für das Verständnis der Programmiersprache GPL waren mir sowohl das Buch von H. Martin, „TI99/4A Intern", als auch die Serie über GPL in der TI-Revue von S. Dyroff eine große Hilfe.
Das Programm versteht und übersetzt sämtliche GPL Befehle und -Formate der Operanden. Der GLP-Befehl FMT wird nebst allen Unterbefehlen dekodiert. Die Ausgabe erfolgt auf den Bildschirm (Text-Mode), bzw. auf ein Diskettenfile im Format Display, Variable 80. Bei der Ausgabe wird zuerst die jeweilige Adresse, dann der Befehlscode und danach die etwaigen Operanden angegeben.
Der GPL-Befehl FMT wird in folgender Weise in seine Unterbefehle dekodiert:
- HPRT Text
  Horizontale Darstellung eines max. 32stelligen Textes
- VPRT Text
  Vertikale Darstellung eines max. 32stelligen Textes
- HPRT CHAR:>4B (K) TIMES 4
  Horizontale Darstellung eines Zeichens (hier z.B. der Character K mit Hex.-Code>4B) mehrmals wiederholt (hier z.B. 4mal).
- VPRT CHAR:>4B (K) TIMES 4
  Wie zuvor, nur vertikale Darstellung
- YPT RELATIVE YPT=YPT+>1F
  Relative Neubestimmung des Zeilenpointers (>837E) um einen bestimmten Wert (hier z.B. >1F).
- XPT RELATIVE XPT=XPT+>1F
  Wie zuvor, nur wird hier der Spaltenpointer (>837F) neubestimmt.
- LOOP 1 FOR 1 TO 5
  Start einer Schleife innerhalb eines FMT-Befehles. Angegeben wird Schleifen-Nr. (hier z.B. 1) und Anzahl der Durchläufe (hier z.B. 5). Auch verschachtelte Schleifen werden vom Programm erkannt und hiermit angezeigt.
- END LOOP 1
  Ende der Schleife mit der entsprechenden Schleifennummer (hier z.B. Nr. 1).
- OFFSET=>60
  Setzen des Screenoffsets auf Absolutwert (hier z.B. >60)
- XPT=>1A
  Absoluten Wert an Spaltenpointer übergeben (hier z.B. >1A)
- YPT=>1A
  Absoluten Wert an Zeilenpointer übergeben (hier z.B. >1A)
- OFFSET=*>835E
  Screenoffset wird auf Wert aus Speicherstelle (hier z.B. >835E) gesetzt.
- 3*PRT BYTE FROM *GROM>2000
  Bringe mehrere Zeichen (hier z.B. 3) aus Speicherstelle (hier z.B. GROM>2000) und folgenden Speicherstellen auf den Bildschirm.
- END FMT
  Beendigung des FMT-Befehls.

Ich habe diese ausführliche Darstellung der FMT-Unterbefehle derjenigen in dem Buch von H. Martin vorgezogen, da sie leichter zu verstehen ist.

Bedienung des Programms:
Nach dem Starten des Programmes meldet es sich mit dem Titelbild (Hauptmenü). Hier wird man aufgefordert, folgende 3 Fragen zu beantworten:
1.) Ist eine Ausgabe als GPL-Code gewünscht oder sollen die Werte als Data-Segments angegeben werden? Letztere Möglichkeit gibt jeweils zwei aufeinanderfolgende Bytes in der Form aus:
Adresse DATA Adresseninhalt TEXT Ausgabe der Bytes als ASCII
Bei der Ausgabe der ASCII-Zeichen werden von den Bytes, falls die Werte größer als 128 sind, der Basic-Offset von 96 subtrahiert. Bei Werten kleiner als 32 wird ein Sternchen geschrieben (*).
2.) Soll ein GROM dekodiert werden oder eine Kopie eines GROM-Inhalts? Diese Kopie muß sich ab Speicherstelle E800 (hex) im RAM befinden. Es stehen einem hier 6 kB frei zur Verfügung. Bei der Übersetzung aus diesem Speicherbereich heraus sind bei der folgenden Frage die wahren GROM-Adressen einzugeben, da sonst die Werte bei den GPL-Befehlen B und CALL nicht mit den angegebenen Speicheradressen übereinstimmen.
Beide bisherigen Fragen sind durch Tastendruck (Taste 1 oder 2) zu beantworten.
3.) Nun erfolgt die Eingabe des Speicherbereiches, der übersetzt werden soll. Sie muß in hexadezimaler

```
/***************************************
** GPL-DISASSEMBLER V1.3 VOM MAI '87  **
****************************************
** c by Klaus-Dieter Oppermann        **
***************************************/
#asm
       REF  PRINTF,C$GPLL,VMBR
#endasm
#include "DSK1.STDIO"
int grom,zeile,spalte,hilf,raro,cd;
char form1[]="%04x  D%-6s",
     form2[]="%04x  %-7s";
/**************************************/
main()
{ int adr1,adr2,hi; char byte,k;
while (1)
{ grom=cd=raro=0;
  titel();
  while (1)
  { locate(16,17); beep(); adr1=input();
    locate(18,17); beep(); adr2=input();
    if (adr1<adr2) break;
  }
grom=check(adr1);
menue();
while(1)
{ byte=peekg(adr1,byte);
  if (cd) adr1=daten(adr1); else adr1=test(adr1,byte);
  if (adr1>=adr2) { keybrd(); break; }
  if (zeile>=23) { if ((h1=keybrd())==1) break; else menue(); }
} } }
/**************************************/
input()
{ char k; int i,j,ad;
i=4; ad=0;
while(i)
  { k=getchar();
    if (k==10) { for (j=1;j<=24;j++) putchar(8); i=4; continue; }
    else if ((k>47)&(k<58)) k=k-48;
    else if ((k>64)&(k<71)) k=k-55;
    else if ((k>96)&(k<103)) { putchar(8); putchar(k-32); k=k-87; }
    else { putchar(8); continue; }
    ad=ad<<4;
    ad=ad+k; i--;
  }
return(ad);
}
/**************************************/
peekg(adr,byte) int adr; char *byte;
{
if (raro==1)
{ adr=adr-6144-(grom*8192);
#asm
        MOV  @4(14),0
        MOVB *0,@3(14)
#endasm
}
else
{
#asm
        MOV  @4(14),0
        MOVB 0,@>9C02
        SWPB 0
        MOVB 0,@>9C02
        SWPB 0
        MOVB @>9800,@3(14)
```

Schreibweise erfolgen. Es sind immer 4 Ziffern einzutasten und ungültige Eingaben werden ignoriert. Sollte die Endadresse kleiner sein als die Startadresse,

```
#endasm
}
return(byte); }
/***************************************/
titel()
{
putchar(12); stern();
locate(4,9);
puts("GPL - DISASSEMBLER  V1.3");
locate(7,1); stern();
quest();
locate(16,1);
puts("START-ADRESSE =\n\n");
puts(" STOP-ADRESSE =\n\n\n");
puts("Datendiskette in Laufwerk 1\n\n");
stern(); }
/***** Funktion beep() ****************/
#asm
BEEP    BL @C$GPLL
        DATA >34
        B *13
#endasm
/***************************************/
check(v) int v;
{
v=v&(-4096); v=v>>12;
v=v&15; v=v>>1;
return(v);
}
/***************************************/
menue()
{
putchar(12);
zeile=spalte=1;
locate(23,1);
stern();
locate(24,3);
puts("N)ext   Q)uit   R)edo   P)rint");
}
/*********************  *********  ****/
keybrd()
{ int k;
k=0; beep();
while ((k!='Q')&(k!='R')&(k!='N')&(k!='P'))
 { locate(24,39);
   k=getchar();
   if (k>96) k=k-32;
 }
if (k=='P') druck();
putchar(12);
if (k=='Q') exit(0);
if (k=='R') return(1);
return(0);
}
/***************************************/
setaus()
{ zeile++; spalte=1; }
/***************************************/
test(ad,by) int ad; char by;
{ hilf=0;
switch (by) {
case 0: typ3(ad,"RTN"); break;
case 1: typ3(ad,"RTNC"); break;
case 3: typ3(ad,"SCAN"); break;
case 9: typ3(ad,"H"); break;
case 10: typ3(ad,"GT"); break;
```

so muß neu eingegeben werden. Vor Beendigung der jeweiligen Eingabe kann, durch Betätigen der Enter-Taste, die bisherige Eingabe gelöscht und neu eingetastet werden.

```
case 11: typ3(ad,"EXIT"); break;
case 12: typ3(ad,"CARRY"); break;
case 13: typ3(ad,"OVF"); break;
case 16: typ3(ad,"CONT"); break;
case 17: typ3(ad,"EXEC"); break;
case 18: typ3(ad,"RTNB"); break;
case 19: typ3(ad,"RTGR"); break;
case 2: ad=typ21(ad,"RAND"); break;
case 4: ad=typ21(ad,"BACK"); break;
case 7: ad=typ21(ad,"ALL"); break;
case 14: ad=typ21(ad,"PARSE"); break;
case 15: ad=typ21(ad,"XML"); break;
case 5: ad=typ22(ad,"B"); break;
case 6: ad=typ22(ad,"CALL"); break;
case 8: ad=typfmt(ad); break;
default: break; }
if (hilf) { ad++; return(ad); }
while (1) {
if ((by>19 & by<32) | (by>151 & by<160)) { typ3(ad,"XGPL"); break; }
if ((by>239 & by<244) | (by>251 & by<=255)) { typ3(ad,"XGPL"); break; }
if (by>=32 & by<=63) { ad=typ6(ad,by); break; }
if (by>=64 & by<=95) { ad=typ4(ad,"BR",by); break; }
if (by>=96 & by<=127) { ad=typ4(ad,"BS",by); break; }
if (by==128 | by==129) { ad=typ5(ad,"ABS",by); break; }
if (by==130 | by==131) { ad=typ5(ad,"NEG",by); break; }
if (by==132 | by==133) { ad=typ5(ad,"INV",by); break; }
if (by==134 | by==135) { ad=typ5(ad,"CLR",by); break; }
if (by==136 | by==137) { ad=typ5(ad,"FETCH",by); break; }
if (by==138 | by==139) { ad=typ5(ad,"CASE",by); break; }
if (by==140 | by==141) { ad=typ5(ad,"PUSH",by); break; }
if (by==142 | by==143) { ad=typ5(ad,"CZ",by); break; }
if (by==144 | by==145) { ad=typ5(ad,"INC",by); break; }
if (by==146 | by==147) { ad=typ5(ad,"DEC",by); break; }
if (by==148 | by==149) { ad=typ5(ad,"INCT",by); break; }
if (by==150 | by==151) { ad=typ5(ad,"DECT",by); break; }
if (by >= 248) { ad=typ1(ad,"SWGR",by); break; }
if (by >= 244) { ad=typ1(ad,"I/O",by); break; }
if (by >= 236) { ad=typ1(ad,"COINC",by); break; }
if (by >= 232) { ad=typ1(ad,"SRC",by); break; }
if (by >= 228) { ad=typ1(ad,"SRL",by); break; }
if (by >= 224) { ad=typ1(ad,"SLL",by); break; }
if (by >= 220) { ad=typ1(ad,"SRA",by); break; }
if (by >= 216) { ad=typ1(ad,"CLOG",by); break; }
if (by >= 212) { ad=typ1(ad,"CEQ",by); break; }
if (by >= 208) { ad=typ1(ad,"CGE",by); break; }
if (by >= 204) { ad=typ1(ad,"CGT",by); break; }
if (by >= 200) { ad=typ1(ad,"CHE",by); break; }
if (by >= 196) { ad=typ1(ad,"CH",by); break; }
if (by >= 192) { ad=typ1(ad,"EX",by); break; }
if (by >= 188) { ad=typ1(ad,"ST",by); break; }
if (by >= 184) { ad=typ1(ad,"XOR",by); break; }
if (by >= 180) { ad=typ1(ad,"OR",by); break; }
if (by >= 176) { ad=typ1(ad,"AND",by); break; }
if (by >= 172) { ad=typ1(ad,"DIV",by); break; }
if (by >= 168) { ad=typ1(ad,"MUL",by); break; }
if (by >= 164) { ad=typ1(ad,"SUB",by); break; }
if (by >= 160) { ad=typ1(ad,"ADD",by); break; }
}
ad++; return(ad); }
/*****************************************/
typ1(a,c,b) int a; char *c,b;
{ int j; char op;
j=0;
loczs();
if (b&1) printf(form1,a,c);
else printf(form2,a,c);
```

```
spalte=14; a++; op=peekg(a,op);
a=operand(op,a);
a++;
op=peekg(a,op); loczs();
if (b&2) { j=op;
           if (b&1) { a++; op=peekg(a,op); j=j<<8; j=j+op; printf(",>%04x",j); }
           else printf(",>%02x",j);
           }
else       { putchar(44); spalte++; a=operand(op,a);
           }
setaus(); return(a);
}
/**************************************/
typ21(a,c) int a; char *c;
{ char im;
hilf=1;
loczs(); printf(form2,a,c);
a++; im=peekg(a,im); printf(">%02x",im);
setaus(); return(a);
}
/**************************************/
typ22(a,c) int a; char *c;
{ int k; char im;
hilf=1; k=0; loczs(); printf(form2,a,c);
a++; im=peekg(a,im); k=im; k=k<<8;
a++; im=peekg(a,im); k=k+im;
printf("GROM@>%04x",k);
setaus(); return(a);
}
/**************************************/
typ3(a,c) int a; char *c;
{ hilf=1;
loczs(); printf(form2,a,c);
setaus();
}
/**************************************/
typ4(a,c,b) int a; char *c,b;
{ char h; int k;
h=31; k=0; loczs(); printf(form2,a,c);
k=b&h; k=k<<8; k=k+grom*8192; a++;
h=peekg(a,h); k=k+h;
printf("GROM@>%04x",k);
setaus(); return(a); }
/**************************************/
typ5(a,c,b) int a; char *c,b;
{  char senk;
loczs();
if (b&1)  printf(form1,a,c);
else printf(form2,a,c);
a++; senk=peekg(a,senk); spalte=14;
a=operand(senk,a);
setaus(); return(a); }
/**************************************/
typ6(a,b) int a;  char b;
{ int hi;  char ci;
if (zeile==22) { setaus(); a--; return(a); }
hi=0;
loczs();
printf(form2,a,"MOVE");
spalte=14;
a++; ci=peekg(a,ci);
if ((b&1)==1)
      { hi=ci; hi=hi<<8; a++;
        ci=peekg(a,ci); hi=hi+ci;
        loczs(); printf(">%04x",hi);
        spalte=spalte+6;
```

Nach Beantwortung dieser Fragen erfolgt sofort die Ausgabe in der gewählten Form.
Ist ein Bildschirm mit Zeichen gefüllt, stehen einem folgende Wahlmöglichkeiten zur Verfügung:

```
    }
else { a=operand(ci,a); spalte++; }
loczs(); puts("TO "); spalte=spalte+3;
a++; ci=peekg(a,ci);
if ((b&16)==0) a=typ6a(a,"GRAM",ci);
else if ((b&8)==8) { loczs(); printf("REG>%02x",ci); }
else a=operand(ci,a);
a++; ci=peekg(a,ci);
zeile++;
locate(zeile,14); puts("FROM "); spalte=19;
if ((b&4)==4) { a=operand(ci,a); setaus(); return(a); }
a=typ6a(a,"GROM",ci);
if ((b&2)==2) { a++; ci=peekg(a,ci); hi=33536+ci;
                spalte=spalte+10; loczs();
                printf("(@>%04x)",hi); }
setaus(); return(a);
}
/**************************************/
typ6a(a,c,b) int a; char *c,b;
{ int h;
h=0; h=b; h=h<<8;
a++; b=peekg(a,b);
h=h+b;
loczs();
printf("%4s@>%04x",c,h);
return(a); }
/**************************************/
typfmt(a) int a;
{ int i,lp,reg; char c,ci;
hilf=1; lp=0;
if (zeile==22) { setaus(); a--; return(a); }
loczs(); printf("%04x  FMT",a);
while(1)
{ setaus();
  if (zeile>=23) { keybrd(); menue(); }
  a++; c=peekg(a,c); ci=c&224;
  reg=0; reg=(c&31)+1; spalte=spalte+2;
  loczs();
 if (ci==0) { puts("HPRT"); putchar(34);
              for (i=1;i<=reg;i++) { a++; c=peekg(a,c);
                                      c=asctst(c); putchar(c); }
              putchar(34); continue; }
 if (ci==32) { puts("VPRT"); putchar(34);
               for (i=1;i<=reg;i++) { a++; c=peekg(a,c);
                                       c=asctst(c); putchar(c); }
              putchar(34); continue; }
 if (ci==64) { a++; c=peekg(a,c); ci=asctst(c);
               printf("HPRT CHAR:%02x (%c) TIMES %2d",c,ci,reg);
               continue; }
 if (ci==96) { a++; c=peekg(a,c); ci=asctst(c);
               printf("VPRT CHAR:%02x (%c) TIMES %2d",c,ci,reg);
               continue; }
 if (ci==128) { printf("YPT RELATIVE YPT=YPT+>%02x",reg); continue; }
 if (ci==160) { printf("XPT RELATIVE XPT=XPT+>%02x",reg); continue; }
 if (ci==192) { lp++; printf("LOOP%2d  FOR 1 TO %2d",lp,reg); continue; }
 if ((c==251)&(lp!=0)) { a=a+2; printf("END LOOP%2d",lp); lp--; continue; }
 if (c==251) { puts("END FMT"); setaus(); return(a); }
 a++; ci=peekg(a,ci);
 if (reg==29) { printf("OFFSET=>%02x",ci); continue; }
 if (reg==31) { printf("YPT=>%02x",ci); continue; }
 if (reg==32) { printf("XPT=>%02x",ci); continue; }
 if (reg==30) { puts("OFFSET=*"); spalte=11; }
 else         { printf("%2d*PRT BYTE FROM *",reg); spalte=21; }
 a=operand(ci,a);
} }
/**************************************/
```

```
operand(b,a) char b; int a;
{ int h83,hi; char c1,c2;
hi=0; h83=33536;
if ((b&128)==0) { hi=b+h83; loczs(); printf("@>%04x",hi);
                  spalte=spalte+6; return(a); }
a++;
if ((b&32)==32) { puts("VDP"); spalte=spalte+3; h83=0; }
if ((b&16)==16) { putchar(42); spalte++; h83=33536; }
else { putchar(64); spalte++; }
if ((b&15)==15)
 { c1=peekg(a,c1); a++;
   c2=peekg(a,c2);
 }
else
 { c1=b&15; c2=peekg(a,c2); }
hi=c1; hi=hi<<8; hi=hi+c2+h83;
loczs();
printf(">%04x",hi); spalte=spalte+5;
if ((b&64)==64)
 { a++;
   c1=peekg(a,c1); hi=0;
   hi=33536+c1;
   if ((spalte+7)>40) { zeile++; spalte=14; }
   loczs();
   printf("(@>%04x)",hi);
   spalte=spalte+8;
 }
return(a);
}
/************************************/
druck()
{ char b[42]; int z,unit,adr;
unit=fopen("DSK1.GPLDATA","a80");
b[41]=0;
for (z=0;z<=22;z++)
{
 adr=z*40;
 peekv(adr,b);
 fputs(b,unit);
}
fclose(unit);
}
/*********************************/
peekv(a,b) int a,*b;
{
#asm
      MOV @4(14),0
      MOV @2(14),1
      LI 2,40
      BLWP @VMBR
#endasm
}
/*********************************/
daten(a) int a;
{  char c,c1,b,b1;  int h;
h=a; locate(zeile,1);
c=peekg(a,c); b=asctst(c);
a++;
c1=peekg(a,c1); b1=asctst(c1);
printf("%04x   DATA    >%02x%02x      TEXT %c%c",h,c,c1,b,b1);
a++;
zeile++;
return(a);
}
/*************************************/
asctst(ch) char ch;
```

SOUNDABLAUF OHNE ZEITVERLUST

In TI-REVUE 5/'87 brachten Sie den Artikel „Der Ton macht die Musik". Am Ende dieses Artikels wurde kurz angeführt, wie man eine Interruptgesteuerte Soundliste aufbaut. Dies ist aber sehr kompliziert und aufwendig. Nun meine Frage. Gibt es ein Programm, das einem die Arbeit abnimmt und eine solche Soundliste aufbaut? Kann man dann den Sound *während* dem Ablauf eines Programmes ablaufen lassen, ohne daß dies eine Unterbrechung des Programmes bedeutet?
Heinz Schehl, Mannheim

Ihre Frage ist sehr schnell beantwortet. Ein solches Programm wurde in TI-REVUE „Assembler-Spezial" veröffentlicht. Der Name ist „Interruptgesteuerte Soundlisten-verarbeitung"!

```
{
if (ch<32) return('*');
if ((ch>127)&(ch<224)) ch=ch-96;
return(ch);
}
/**************************************/
frage(z,s) int z,s;
{ int k;
k=0; beep();
while ((k!='1') & (k!='2')) { locate(z,s); k=getchar(); }
return(k-'1');
}
/**************************************/
stern()
{ int i;
for (i=1;i<=40;i++) putchar(42);
}
/**************************************/
loczs()
{ locate(zeile,spalte); }
/**************************************/
quest()
{ locate(10,1); puts("PRESS  1 fuer GPLCODE");
  locate(11,8); puts("2 fuer GPLDATA");
cd=frage(11,26);
locate(13,1); puts("PRESS  1 fuer POKE-GROM");
locate(14,8); puts("2 fuer POKE-RAM (>E800)");
raro=frage(14,35);
}
```

- Next (Taste N) Es wird die Dekodierung fortgesetzt. Ist man in dem zu dekodierenden Bereich am Ende angelangt, hat diese Taste dieselbe Wirkung wie die Taste R.
- Redo (Taste R) Hiermit wird zum Titelbild des Programmes zurückgekehrt.
- Quit (Taste Q) Nach Drücken dieser Taste wird über "exit(0);" das Programm verlassen. Betätigt man danach die Taste Y so landet man wieder im Hauptmenü des Programms.
- Print (Taste P) Vor Drücken dieser Taste ist es notwendig, in Laufwerk 1 eine Diskette einzulegen. Auf ihr wird eine Datei vom Typ Display, Variable 80, Append angelegt, der Filename lautet: GPLDATA.
  Dieses File läßt sich dann z.B mit dem Editor des E/A-Moduls weiterbearbeiten und mit der Print-Option des Moduls ausdrukken. Nach Beenden der Ausgabe des Bildschirminhalts auf Diskette wird automatisch der Next-Befehl (siehe dort) ausgeführt.

Innerhalb der Übersetzung, d.h. vor Beendigung, eines FMT-Befehls sind nur die Tasten N, P und Q aktiv. Ein Redo ist nur möglich, über die Tastenfolge Q und dann Y.

Anmerkung:
Die Dekodierung von GPL ist leider nicht so einfach, da praktisch jedes Byte einen Befehl darstellt. Aufgrund dessen können Datenwerte (DATA) als solche nicht erkannt werden. Unsinnige Befehlsausgaben erkennt man häufig an unmöglichen GROM- bzw. GRAM-Zugriffen oder an nicht enden wollenden FMT-Ausgaben. Letztere sind nur über die Taste Q zu stoppen, mit Neustart des Programmes über Y.

IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Geschäftsführer: Werner E. Seibt
Redaktion: Alfons Bauer, Thomas Dehn, Jürgen Puschmann, Uwe Schröder
Verantwortlich für den Inhalt:
MSX-Teil: Werner E. Seibt
TI-Teil: Oliver Huber (Technik) Hartmut König (Listings)
Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d
Alle: Postfach 1161
8044 Lohhof.
Anfragen bitte nur schriftlich
Druck: Dierichs, Kassel
Es gilt die Honorarliste des Verlages.

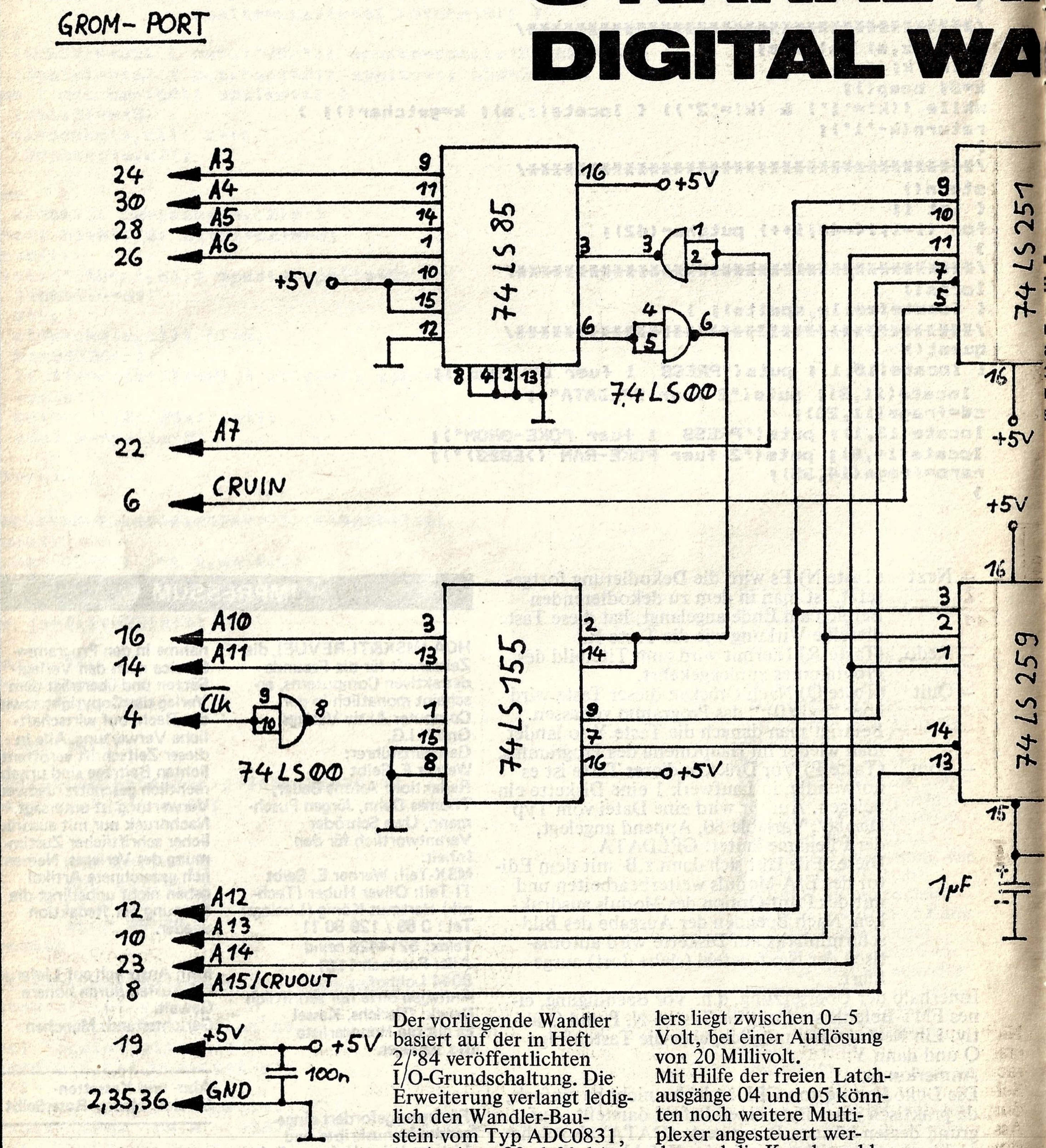
Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.

Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

©by CA Verlag, Heßstraße 90, 8000 München 40 SPS und Autoren

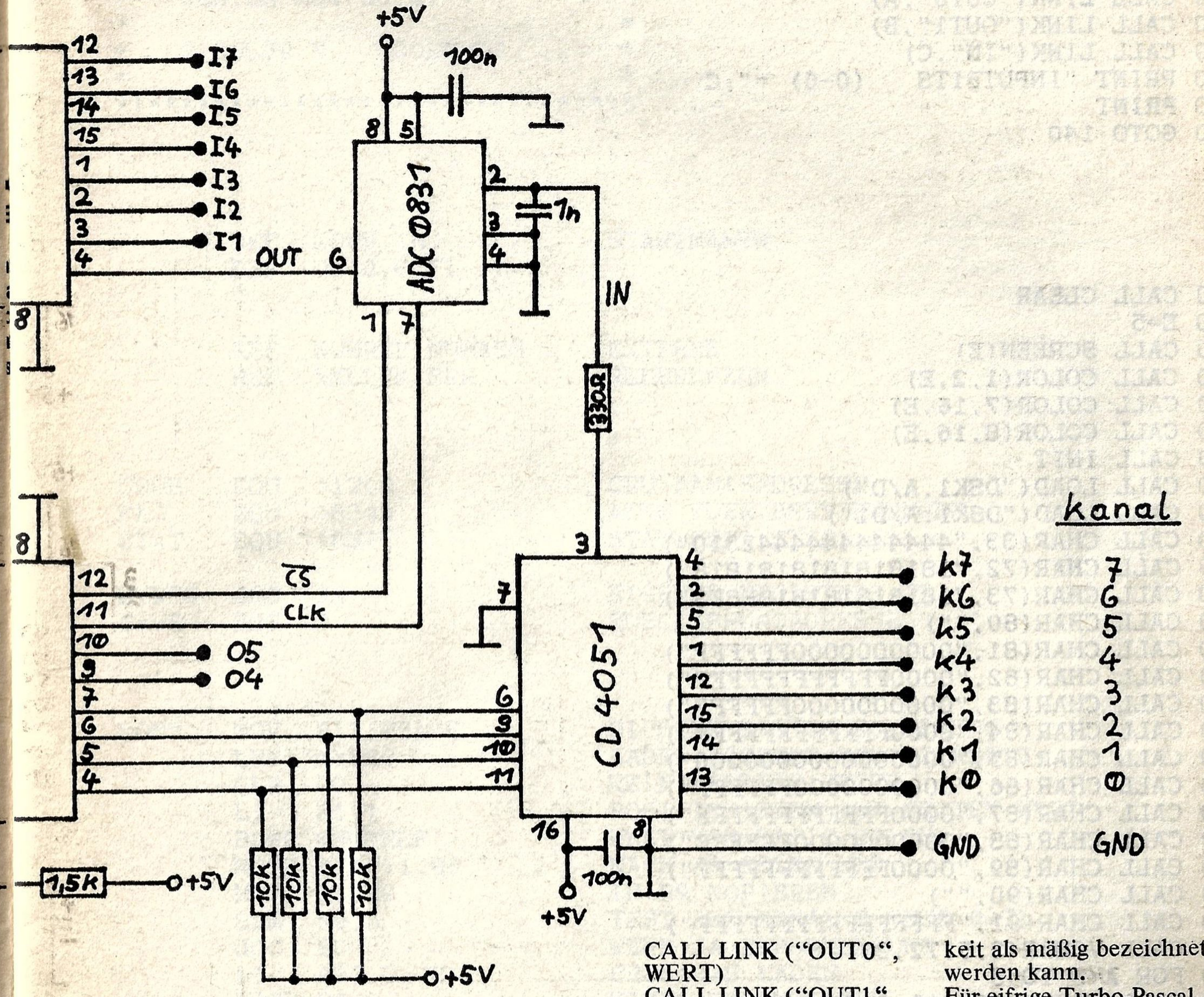
8 KANAL A
DIGITAL WA



Der vorliegende Wandler basiert auf der in Heft 2/'84 veröffentlichten I/O-Grundschaltung. Die Erweiterung verlangt lediglich den Wandler-Baustein vom Typ ADC0831, sowie einen 8fach-Multiplexer CD4051. Der Bereich des Wandlers liegt zwischen 0–5 Volt, bei einer Auflösung von 20 Millivolt.

Mit Hilfe der freien Latchausgänge 04 und 05 könnten noch weitere Multiplexer angesteuert werden und die Kanalanzahl somit vervielfacht werden.

NALOG / NDLER



Nicht aus dem Schaltplan ersichtlich dürfte die Tatsache sein, daß diese Schaltung nur in Verbindung mit dem Editor/Assembler-Modul funktioniert. Siehe dazu Anleitung Heft 2/'84.

Folgende Maschinen-Unterprogramme werden zur Verfügung gestellt:

CALL LINK ("KONV", KANAL,WERT) startet die A/D-Wandlung Eingabe für Kanal 0 – 7 Ergebnis Wert 0.00 – 5.00 Volt

CALL LINK ("IN", WERT) liest die Bits (I1 – I7) in das System ein

CALL LINK ("OUT0", WERT)

CALL LINK ("OUT1", Wert)

Gültige Eingaben für Wert sind 0 (gelöscht) oder 1 (Bit gesetzt)

OUT0 setzt oder löscht 04

OUT1 setzt oder löscht 05

Das Basicdemo Programm stellt nur ein einfaches Beispiel zur graphischen Darstellung dar, dessen Geschwindigkeit als mäßig bezeichnet werden kann.

Für eifrige Turbo-Pascal-Programmierer habe ich daher die notwendige Procedure KONV geschrieben sowie die Anpassung des Maschinenprogramms vorgenommen.

Somit steht einer schnellen grafischen Aufbereitung der eingelesenen 8 Kanäle nichts mehr entgegen.

Reiner Ziegler

```
100 CALL CLEAR
110 CALL INIT
120 CALL LOAD("DSK1.A/D")
130 CALL LOAD("DSK1.A/D1")
140 INPUT "OUTPUTBIT 0 (1/0) = ":A
150 PRINT
160 INPUT "OUTPUTBIT 1 (1/0) = ":B
170 PRINT
180 CALL LINK("OUT0",A)
190 CALL LINK("OUT1",B)
200 CALL LINK("IN",C)
210 PRINT "INPUTBITS   (0-6) =";C
220 PRINT
230 GOTO 140
```

```
100 CALL CLEAR
110 E=5
120 CALL SCREEN(E)
130 CALL COLOR(1,2,E)
140 CALL COLOR(7,16,E)
150 CALL COLOR(8,16,E)
160 CALL INIT
170 CALL LOAD("DSK1.A/D")
180 CALL LOAD("DSK1.A/D1")
190 CALL CHAR(33,"4444444444442810")
200 CALL CHAR(72,"1818181818181818")
210 CALL CHAR(73,"181818181818F8F8")
220 CALL CHAR(80,"")
230 CALL CHAR(81,"0000000000FFFFFF")
240 CALL CHAR(82,"0000FFFFFFFFFFFF")
250 CALL CHAR(83,"0000000000FFFFFF")
260 CALL CHAR(84,"0000FFFFFFFFFFFF")
270 CALL CHAR(85,"0000000000000000")
280 CALL CHAR(86,"0000000000FFFFFF")
290 CALL CHAR(87,"0000FFFFFFFFFFFF")
300 CALL CHAR(88,"0000000000FFFFFF")
310 CALL CHAR(89,"0000FFFFFFFFFFFF")
320 CALL CHAR(90,"")
330 CALL CHAR(91,"FFFFFFFFFFFFFFFF")
340 CALL VCHAR(2,5,72,20)
350 FOR A=0 TO 5
360 CALL HCHAR(22-4*A,3,48+A)
370 CALL HCHAR(22-4*A,4,33)
380 CALL HCHAR(22-4*A,5,73)
390 NEXT A
400 PRINT "Kanal 0  1  2  3  4  5  6  7"
410 FOR B=0 TO 7
420 CALL LINK("KONV",B,C)
430 IF C<=5.1 THEN 450
440 C=5.1
450 D=(22-INT(ABS(C)*4))
460 CALL VCHAR(D,8+3*B,91,22-D)
470 CALL VCHAR(D,9+3*B,91,22-D)
480 C=INT((C-INT(C))*10)
490 CALL VCHAR(1,8+3*B,32,D-2)
500 CALL VCHAR(1,9+3*B,32,D-2)
510 CALL VCHAR(D-1,8+3*B,80+C)
520 CALL VCHAR(D-1,9+3*B,80+C)
530 NEXT B
540 GOTO 410
```

```
**********************************
*                                *
*   ANALOG/DIGITAL - WANDLER     *
*                                *
*   MIT INPUT/OUTPUT KANAELEN    *
*                                *
*   DESIGN MAI 86 BY             *
*                                *
*   ZIEGLER R.  MOOSBURG         *
*                                *
**********************************


        DEF  KONV,IN          STARTNAMEN
        DEF  OUT0,OUT1

        REF  NUMREF,NUMASG    EXTERNE
        REF  XMLLNK,ERR       REFERENZEN


CRUB    EQU  >1200            CRU BASISADRESSE
FAC     EQU  >834A            AKKU FUER BERECHNUNGEN
STAT    EQU  >837C            STATUSREGISTER

MYREG   BSS  32               EIGENER REGISTERSATZ
SAVE    BSS  2                RUECKSPRUNGADRESSE


KONV    MOV  R11,@SAVE        SICHERN BASICRUECKSPRUNG
        LWPI MYREG            LADEN EIGENER REGISTERSATZ
        CLR  R0               KEIN FELDPARAMETER
        LI   R1,1             POSITIONIEREN AUF ERSTEN WERT
        BLWP @NUMREF          WERT VON BASIC HOLEN
        MOVB @FAC+1,R6        NACH R6 HOLEN UND
        MOV  R6,R9            IN R9 KOPIEREN
        SLA  R9,5             TEST AUF KANAL 8 (NUR 0-7)
        JOC  EC               WENN JA ERRORMELDUNG
        LI   R12,CRUB         CRUBASIS LADEN
        LDCR R6,3             KANAL SELEKTIEREN
        SBO  7                WANDLERBAUSTEIN FUER
        SBZ  6                MESSUNG VORBEREITEN
        SBZ  3                MULTIPLEXER FREIGEBEN

        LI   R5,9             SCHLEIFENZAEHLER LADEN
        SBZ  7                WANDLER FREIGEBEN
        CLR  R8               ERGEBNISREGISTER LOESCHEN
CLOCK   SBO  6                WANDLER MIT
        NOP                   EINEM CLOCKIMPULS
        NOP                   ANSTEUERN
        SBZ  6
```

```
        TB   0              AUSGABESIGNAL (ZUERST MSB)
        JNE  COUNT          TESTEN
        ORI  R8,>0001       AUSGABEWERT NACH R8 SICHERN
COUNT   SLA  R8,1           NAECHSTE MESSUNG VORBEREITEN
        DEC  R5             SCHLEIFENZAEHLER VERRINGERN
        JNE  CLOCK          NAECHSTER MESSWERT
        SBO  7              WANDLER SPERREN
        SBO  3              MULTIPLEXER SPERREN

        MOV  R8,@FAC        ERGEBNIS NACH FAC
        BLWP @XMLLNK        IN RADIX 100 FORMAT
        DATA >2300          UMWANDELN
        LI   R9,>0100       FORMATIEREN ZU
        SB   R9,@FAC        0.00 - 5.00
        CLR  R0             KEIN FELDPARAMETER
        LI   R1,2           POSITION AUF ZWEITEN WERT
AUSG    MOVB R0,@STAT       STATUSREGISTER LOESCHEN
        BLWP @NUMASG        WERT AN DAS BASICPROGRAM
        MOV  @SAVE,R11      RUECKSPRUNG HOLEN
        B    *R11           ZURUECK INS BASIC

IN      MOV  R11,@SAVE      R11 SICHERN
        LWPI MYREG          REGISTERSATZ LADEN
        LI   R12,CRUB       CRUBASIS LADEN
        CLR  R8             LOESCHE REGISTER
        STCR R8,8           8 BIT EINLESEN
        SRL  R8,1           BIT 0 AUSBLENDEN
        ANDI R8,>FF00       LSB AUSBLENDEN
        SWPB R8             WERT AN DIE RICHTIGE POSITION
        MOV  R8,@FAC        ERGEBNIS NACH FAC
        BLWP @XMLLNK        IN RADIX 100 FORMAT
        DATA >2300          UMWANDELN
        CLR  R0             KEIN FELDPARAMETER
        LI   R1,1           AUF ERSTEN WERT POSITIONIEREN
        JMP  AUSG           WERT AN DAS BASIC UEBERGEBEN

OUT0    MOV  R11,@SAVE      R11 SICHERN
        LWPI MYREG          REGISTERSATZ LADEN
        LI   R12,CRUB       CRUBASIS LADEN
        CLR  R0             KEIN FELDPARAMETER
        LI   R1,1           AUF ERSTEN WERT POSITIONIEREN
        BLWP @NUMREF        WERT AUS BASIC HOLEN
        CLR  R6             R6 LOESCHEN
        MOV  @FAC,R6        NACH R6 LADEN
        CI   R6,>0000       BIT SOLL NULL SEIN
        JNE  O1             ZUR NAECHSTEN ABFRAGE
        SBZ  4              BIT GLEICH NULL
        JMP  O2             ZURUECK INS BASIC
O1      CI   R6,>4001       BIT SOLL EINS SEIN
        JNE  EC             WENN NICHT FEHLEINGABE
        SBO  4              BIT GLEICH EINS
O2      MOV  @SAVE,R11      RUECKSPRUNG HOLEN
        B    *R11           ZURUECK INS BASIC

OUT1    MOV  R11,@SAVE      R11 SICHERN
        LWPI MYREG          REGISTERSATZ LADEN
```

```
       LI    R12,CRUB          CRUBASIS LADEN
       CLR   R0                KEIN FELDPARAMETER
       LI    R1,1              AUF ERSTEN WERT POSITIONIEREN
       BLWP  @NUMREF           WERT AUS BASIC HOLEN
       CLR   R6                R6 LOESCHEN
       MOV   @FAC,R6           NACH R6 LADEN
       CI    R6,>0000          BIT SOLL NULL SEIN
       JNE   03                ZUR NAECHSTEN ABFRAGE
       SBZ   5                 BIT GLEICH NULL
       JMP   04                ZURUECK INS BASIC
03     CI    R6,>4001          BIT SOLL EINS SEIN
       JNE   EC                WENN NICHT FEHLEINGABE
       SBO   5                 BIT GLEICH EINS
04     MOV   @SAVE,R11         RUECKSPRUNG HOLEN
       B     *R11              ZURUECK INS BASIC

EC     LI    R0,>1300          ERRORMELDUNG
       BLWP  @ERR              LADEN UND AUSGEBEN

       END

PROGRAM WANDLER;

    VAR I : INTEGER;
        B : INTEGER;

    PROCEDURE KONV(KANAL : INTEGER; VAR WERT : INTEGER);
      EXTERNAL;

BEGIN

    WRITELN("  A/D WANDLER TEST");

    WHILE TRUE DO BEGIN
    WRITELN;
    WRITE(" WELCHER KANAL ? :  ");
    READ(I);
    KONV(I,B);
    WRITELN;
    WRITE(" SPANNUNG =  ");
    WRITE(B);
    WRITELN;
    END;

END.
```

TI ALS IBM

Von der Firma Triton in San Franzisco ist für 499 Dollar ein Steckkasten auf den Markt gekommen, mit dessen Hilfe der TI IBM-kompatibel wird. Es stehen somit dem TI-User zwei Softwarewelten offen. Will man diesen Steckadapter jedoch besitzen, so benötigt man Beziehungen in den USA. Wie vielleicht einigen schon bekannt, liefert Triton nicht nach Europa. Eventuell findet sich auch eine europäische Firma, die diese Zusatzteile importiert.

Knut Parthey,
Schönau

Vielen Dank für Ihren Hinweis. Es gibt sicherlich einige User, die eine solche Anwendung schon gesucht haben. Die Frage ist nur, inwiefern der TI kompatibel gemacht wird. Es besteht immerhin die Möglichkeit, ihn als weitere Tastatur am IBM zu verwenden. Was man noch an Geräten von IBM zusätzlich braucht, geht aus Ihrem Prospekt leider nicht hervor.

SOUNDAUSGABE OHNE BILDSCHIRM

Ich möchte meinen TI als eine futuristische Art von Türklingel benutzen. Da ich ihn aber nicht gerne als Verstärker im Dauerbetrieb lassen will, ergibt sich folgende Frage: Ist es möglich, die Tonsignale vom TI direkt abzuzapfen, ohne den Umweg Fernseher?
Weiter stellt sich mir die Frage, ob der TI ohne zusätzliche Kühlung ohne weiteres im Dauerbetrieb lauffähig ist?

Kay Sievert,
Hamburg

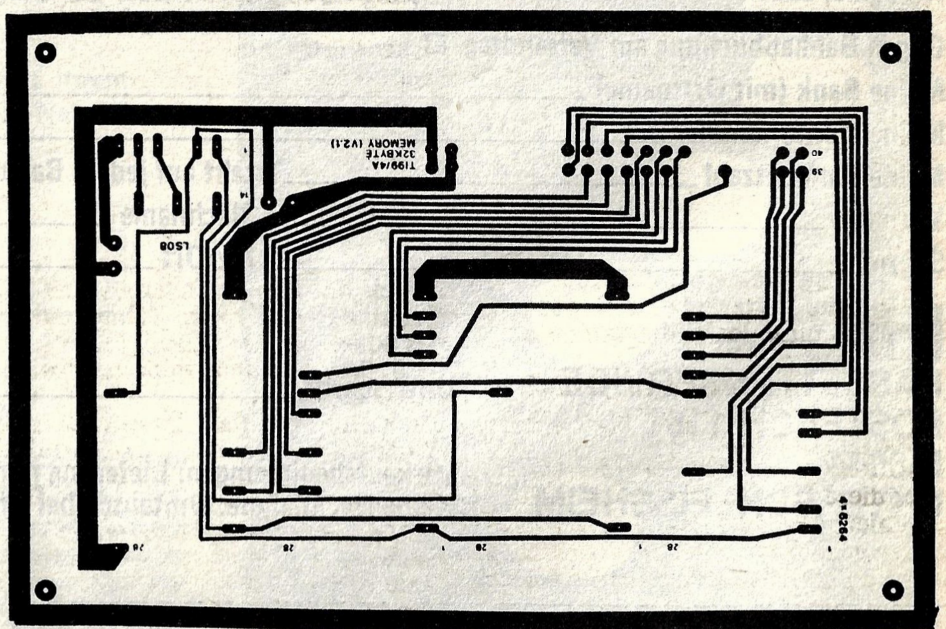
Selbstverständlich ist es möglich, die Tonsignale direkt zu verstärken. Eine Schaltung, die das Tonsignal verstärkt, wurde in TI-REVUE 10/'86 vorgestellt.
Eine Extrakühlung für den Dauerbetrieb sollte man noch für die Spannungsregelung im Computer einbauen. Wie Sie vielleicht schon bemerkt haben, wird die schwarze Fläche beim Modulport nach längerem Betrieb schon beträchtlich warm.

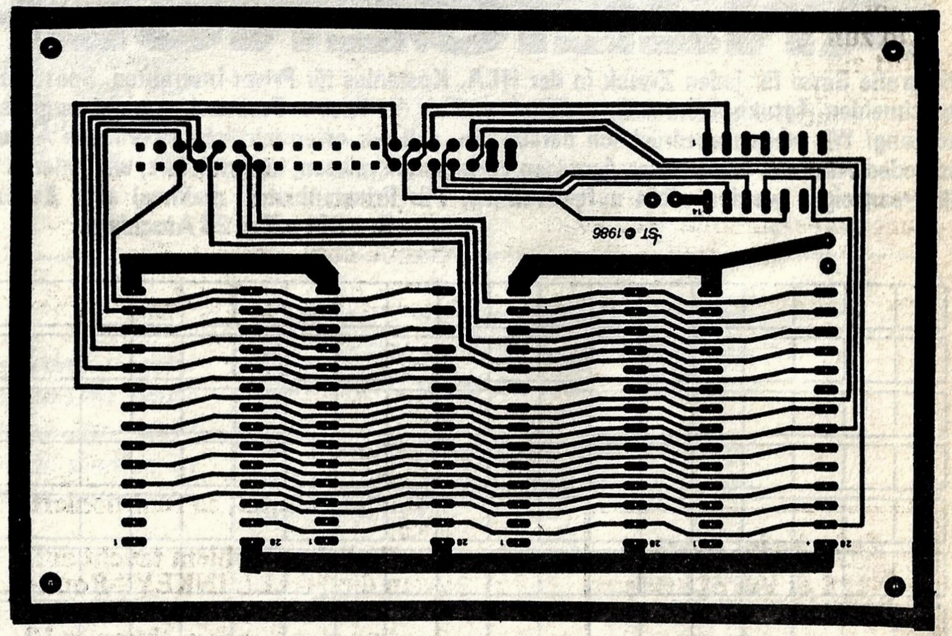
```
*********************************
*                               *
*   ANALOG/DIGITAL - WANDLER    *
*                               *
*   MIT INPUT/OUTPUT KANAELEN   *
*                               *
*   DESIGN MAI 86 BY            *
*                               *
*   ZIEGLER R.   MOOSBURG       *
*                               *
*   VERSION FUER TURBO PASCAL   *
*                               *
*********************************

         DEF  KONV

MYREG    BSS  32                 EIGENER REGISTERSATZ

KONV     LIMI >0000              INTERRUPT AUS
         AI   R0,>FFFC           2 WORTE RESERVIEREN
         LWPI MYREG              REGISTERSATZ LADEN
         MOV  @>8300,R0          POINTER AUF KANAL HOLEN
         MOV  *R0+,R6            KANAL LESEN
         MOV  *R0,R2             POINTER VAR NACH R2
         SWPB R6                 INS MSB SCHIEBEN
         LI   R12,>1200          CRUBASIS LADEN
         LDCR R6,3               KANAL SELEKTIEREN
         SBO  7                  WANDLERBAUSTEIN FUER
         SBZ  6                  MESSUNG VORBEREITEN
         SBZ  3                  MULTIPLEXER FREIGEBEN

         LI   R5,9               SCHLEIFENZAEHLER LADEN
         SBZ  7                  WANDLER FREIGEBEN
         CLR  R8                 ERGEBNISREGISTER LOESCHEN
CLOCK    SBO  6                  WANDLER MIT
         NOP                     EINEM CLOCKIMPULS
         NOP                     ANSTEUERN
         SBZ  6
         TB   0                  AUSGABESIGNAL (ZUERST MSB)
         JNE  COUNT              TESTEN
         ORI  R8,>0001           AUSGABEWERT NACH R8 SICHERN
COUNT    SLA  R8,1               NAECHSTE MESSUNG VORBEREITEN
         DEC  R5                 SCHLEIFENZAEHLER VERRINGERN
         JNE  CLOCK              NAECHSTER MESSWERT
         SBO  7                  WANDLER SPERREN
         SBO  3                  MULTIPLEXER SPERREN

         MOV  R8,*R2             ERGEBNIS ZURUECKGEBEN

         LWPI >8300              WORKSPACE ZURUECKSETZEN
         LIMI >0002              INTERRUPT FREIGEBEN
         B    *R11               ZURUECK INS PASCAL

         END
```

KORREKTUREN

Mit der in der TI-REVUE 7/87 auf Seite 41/52 beschriebenen 32K-Byte Speichererweiterung wird der TI99/4A durch den 16-Bit-Datenbus bis zu doppelt so schnell. Dies deshalb, weil die CPU beim Lesen aus, bzw. beim Schreiben in die Erweiterung nicht mehr über den nach außen geführten und auf 8 Bit verengten Datenbus das Low-Byte und das High-Byte nacheinander lesen bzw. schreiben muß. Die CPU hat so die Möglichkeit, dieses 16-Bit-Wort (MSB + LSB) auf einmal zu lesen bzw zu schreiben. Die doppelte Geschwindigkeit wird somit immer von dieser Erweiterung in den Bereichen (>2000 bis >3FFF und >A000 bis >FFFF) erzielt, da hier durch den direkten Anschluß an den 16-Bit-Datenbus des TMS 9900 die Wartetakte für die CPU wegfallen. Das ROM des TI99/4A (Bereich >0000 – >1FFF), sowie das Scratch-Pad (>8300 – >83FF) in der Konsole sind bereits an diesem 16-Bit-Datenbus angeschlossen und laufen in jedem Fall mit der höheren Geschwindigkeit. Zugriffe auf andere Bereiche, wie VDP, GROMs, Diskettenlaufwerke, RS232, PIO, bleiben genau so schnell (bzw. langsam) wie vorher. Ein Disketten-Kopierprogramm zum Beispiel wird nur etwa 5% schneller, andere Assemblerprogramme, die diese Bereiche wenig oder gar nicht nutzen, laufen, wie schon gesagt, bis zu doppelt so schnell.
Auch alle andere Module, die auf die Erweiterung zugreifen (TI-Writer, Editor/Assembler usw.) werden schneller. Da das Extended Basic ebenfalls das High-Memory benutzt, wird auch hier die Geschwindigkeit hoher, allerdings je nach Programm nur etwa 10 – 20%, da die GROMs die ganze Sache bremsen. In Basic ändert sich gar nichts, da das TI-Basic ja nicht auf die Erweiterung zugreifen kann.
Der Grund, daß einige Module und Programme mit der 16-Bit-Erweiterung fehlerhaft laufen, hat seine Ursache darin, daß der VDP-Prozessor (TMS9929A) dann Schwierigkeiten mit der erhöhten Geschwindigkeit bekommt, wenn auf die VDP-Schnittstellenadressen (>8800, >8802, >8C00, >8C02) zweimal, ohne ein 'NOP' dazwischen, zugegriffen wird. (Siehe auch in dem Buch: „TMS9900 Assembler auf dem TI99/4A“, Seite 115, von Simon Koppelmann.) Bei Nichtbeachtung dieser „Kleinigkeit“ laufen zwar die Programme, aber auf dem Bildschirm sieht es ziemlich wüst aus. Ein Beispiel hierfür sind die Apesoft-Grafik-Programme. Andere Grafik-Programme, wie zum Beispiel der TI-Artist, laufen einwandfrei. Der Fehler liegt also nicht an der Speicher-Erweiterung, sondern daran, daß einige Programmierer dies nicht beachtet haben. Bei den 8-Bit-Memorys wurde dieser 'Nop'-Befehl durch den hardwaremäßigen Wartetakt ersetzt.

Soweit mir bekannt ist, treten mit der 80-Zeichenkarte, die es ja seit einiger Zeit für den TI99/4A gibt, diese Fehler aufgrund des schnelleren Video-Prozessors ebenfalls nicht mehr auf.
Weiterhin hatten Sie bei der vorgeschlagenen Batterie- bzw. Accupufferung zwar eine Zeichnung der Schaltungsänderungen abgedruckt, aber ohne Kommentar ist diese ziemlich unverständlich. Hier deshalb noch einmal meine Beschreibung: Die Speichererweiterung kann bei Bedarf mit einem Accu oder mit einer Batterie gepuffert werden. Auf diese Art bleiben die Daten auch nach dem Ausschalten des Rechners erhalten.





Verkleinerte Layouts

Hierzu sind folgende Änderungen vorzunehmen. Die Spannungsversorgung (+5 Volt) zu den Pins 28 der Speicher ICs wird unterbrochen. Die CS2-Eingänge (Pin 26) allerdings müssen weiterhin direkt an der Spannungsversorgung angeschlossen bleiben (evtl. Brücke ändern). An zusätzlichen Bauteilen werden benötigt:

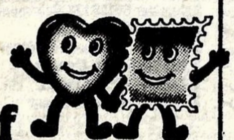
D1 = Siliziumdiode, z.B. 1N4148
D2 = Germaniumdiode, z.B. AA119
Ac = Accu 2,4 bis 3,6 Volt
R1 = Widerstand (Größe nach Accu)
Sollte so groß sein, daß der Ladestrom etwa 10% der Nennkapazität des Accus beträgt. Soll statt des Accus eine Batterie benutzt werden, so muß der Widerstand entfallen.

Die Schaltung wird dann einfach in die unterbrochene Leitung zu den Pins 28 eingefügt.

Sollten Leser Ihrer Zeitschrift sich für diese Schaltung interessieren, so hier meine Adresse und Telefonnummer: Jürgen Stelter, Berliner Straße 2, 6104 Seeheim-Jugenheim, Tel.: 06257/83247. Ich freue mich über jeden, der dem TI99/4A treu bleibt.

Und zum guten Schluß noch ein Wort zur 16-Bit-Erweiterung von R. Ziegler. Drei.ache Geschwindigkeit ist meiner Meinung nach nicht drin. Und der Anschluß der A12-Pins der RAMs an die A2 des Rechners kann ja wohl auch nicht funktionieren. Sollte R. Ziegler anderer Meinung sein, kann er sich ja mal mit mir in Verbindung setzen. Würde mich freuen, von ihm etwas zu hören.

Zeig beim Porto Herz & Verstand:



Kauf Wohlfahrtsbriefmarken

Hilfe, die ihr Ziel erreicht.

Erhältlich bis Ende März bei der Post ganzjährig bei den Wohlfahrtsverbänden

FEHLERHAFTES LISTING IN DER TI-REVUE

In Ihrer letzten Ausgabe der TI-REVUE, der Nummer 7/87, haben Sie ein Listing für Window-managing veröffentlicht (Seite 44/45). Bei Probeläufen sind mir in diesem Listing leider einige Fehler aufgefallen, die nicht auf Tippfehler zurückzuführen sind.

Einer dieser Fehler ist absolut harmlos und betrifft eigentlich nur das Aussehen des Titelbildes, das nach Aufruf von "CALL WINDOW STAR" erscheint.

Die eigentlichen Fehler beziehen sich aber auf das Öffnen und Schließen der Windows. Für diesen Zweck steht unter anderem eine Variable ("AN") zum Zählen der Windows zur Verfügung. Leider wird diese Variable vollkommen falsch verwaltet.

Ein kleines Beispiel: Es werden Windows Nummer 1, 2 und 3 durch "CALL WINDOW" eröffnet. Der Counter "AN' steht dann auch brav auf "3". Jetzt wird Window #1 geschlossen! Durch die Abfrage in Zeile 5250 entsteht ein Fehler. Der Vergleich "NR<AN" ist verkehrt herum gestellt.

Aber selbst wenn man dies ändert, läuft noch nicht alles richtig! Sollte trotzdem (durch unsachgemäße Bedienung) ein Fehler betreffs der Window-Nummern auftreten, so tritt ein Fehler auf, wenn das Fehler-Window wieder geschlossen werden soll. Grund: Wenn nur 3 Windows geöffnet sind, das Programm aber versucht, Window Nummer 10 zu schließen, so kommt es durch die (mittlerweile umgekehrte) Abfrage in Zeile 5250 erneut zu einer Fehlermeldung.

Wenn man aber nicht von vornherein stur Nummer 10 als Fehler-Window nimmt, sondern vielmehr die aktuelle Anzahl plus 1 als Window-Nummer benutzt, so funktioniert alles.

Ein ähnliches Problem taucht auf, wenn die "CALL INKEY"-Routine benutzt wird, da auch diese Routine stur Window Nummer 10 zur Anwendung bringt. Wenn man auch diesen Fehler nach dem voran beschriebenen Verfahren bereinigt, so läuft auch mit diesem Unterprogramm alles korrekt.

Als letztes Problem könnte der Fehler auftreten, daß jemand ein Fenster größer als Nummer 10 eröffnen will. Es kann aber auch passieren, daß schon 10 Windows eröffnet sind, und die "CALL INKEY"- oder die Fehler-Routine Window Nummer 11 eröffnen will. Auch dieser Fehler kann ganz leicht abgefangen werden.

Im folgenden habe ich noch einmal jeweils kurz die Fehler beschrieben und eine Möglichkeit aufgezeigt, wie diese dann zu berichtigen sind. Sind erst einmal alle diese Änderungen vorgenommen worden, so hat man wirklich ein recht gutes Werkzeug zum Verwalten von Windows in der Hand. Vielleicht können sie ja auch einmal bei Gelegenheit mit anderen Routinen angereichert werden?! Ein kleiner Pferdefuß läßt sich aber nicht so ohne weiteres herausnehmen: Wenn die Windows Nummer 1, 2 und 3 geöffnet sind, dann Window Nummer 1 geschlossen wird, so kommt es zu einer Fehlermeldung, wenn dann erst Window Nummer 3 geschlossen werden soll. Grund: Der Counter wird durch Schließen von Nummer 1 auf "2" herabgesetzt. Die einzige Möglichkeit, diesen Fehler zu umgehen, ist, daß man die Windows „von oben nach unten" schließt.

Fehler 1: Titelbild
5160. .::DISPLAY AT(15,15) SIZE (4): "V1 0"

Fehler 2: Verwaltung des Counters "AN"
5250 IF NR>AN THEN 5260 ELSE...

Fehler 3: Schließen von #10 nach Fehler
5260 CALL WINDOW(AN+1, 5,5,18,4,AN)::...
5290.NR=AN :: ...

Fehler 4: Schließen von #10 nach "CALL INKEY"
5300 ... ELSE CALL WINDOW (AN+1,10,10,16,2,AN)
5330 CALL WCLOSE(AN,AN)

Fehler 5: Mehr als 10 Windows
5190 ... :: IF AN>10 THEN SUBEXIT ELSE ADR=768*NR +7424

Genau genommen sind diese Routinen trotz aller Berichtigungen der Fehler immer noch etwas unglücklich programmiert, da das erste Window bereits ab Adresse >2000 (8192) abgespeichert wird. Da aber in diesem Bereich (bis >24F4 bzw. 9460) noch einige wichtige Pointer stehen, kann es zu Interferenzen, z.B. mit Routinen in Assembler, kommen. Ich muß aber zugeben, daß mir auch noch keine bessere Methode eingefallen ist, die sich auch noch einigermaßen einfach realisieren läßt. Es wäre aber z.B. denkbar, die Windows im 24K-Segment abzulegen. Dabei müßte nur überprüft werden, wo das XB-Programm anfängt.

Einige kleinere weitere Schwächen »

werden offenbar, wenn man ein Fenster, in dem schon Text steht, das aber schon von anderen Windows überschrieben wurde, erneut aufrufen will. In den bis jetzt vorliegenden Routinen ist leider nicht vorgesehen, den alten Text erneut auf den Bildschirm zu holen. Wichtig wäre dabei auch noch eine Erweiterung, die es dann auch ermöglicht, in ein Fenster zu schreiben, ohne ganz exakt wissen zu müssen, wo es sich auf dem Bildschirm befindet Ganz davon abgesehen, daß es schon mit einigen Schwierigkeiten verbunden ist, in ein Fenster zu schreiben, das direkt ab Spalte 1 beginnt. Hier muß Text unter Zuhilfenahme von 'HCHAR" oder "VCHAR" dargestellt werden.

RAM-MODUL SCHON SEIT 2 JAHREN BEKANNT

Im vorletzten Heft veröffentlichte R. Ziegler die Schaltung eines RAM-Moduls, mit dem die TI-Module ausgelesen und gefahren werden können. Dazu ist zu sagen, daß wir, der TI-Workshop Wiesbaden, schon seit fast 2 Jahren ein derartiges Modul besitzen. Es ist aber softwaremäßig bereits voll ausgereift und bietet sehr viel mehr Möglichkeiten, als die in HCA beschriebene. Version. Auch andere Workshops haben sich inzwischen derartige Module gebaut, und jeder hat ein anderes Betriebssystem, wobei keines mit den anderen kompatibel ist. Leider wird viel Zeit mit der doppelten und dreifachen Entwicklung ein und derselben Sachen verschwendet. Deshalb sind wir dafür, daß sich die TIler zusammenschließen. Wir z.B. sind ein Workshop, bei dem es keine festen Mitgliedschaften und erst recht keine Mitgliedsbeiträge gibt. Wir sind einfach eine Handvoll Freaks, die Hardware entwickeln und Software tauschen. Wir würden gerne noch mit weiteren TIlern in Verbindung treten. Vor allem Spezialisten für höhere Programmiersprachen wie Forth, UCSD-Pascal, Pilot, Logo fehlen uns.

TI-Workshop Wiesbaden
Eleonorenstraße 6
6200 Wiesbaden

Übrigens: Auf dem TI-Treff in Bremen sind wir selbstverständlich wieder da!!! Wobei wir eine neue 512kB-Ramdisk mit Supersoftware vorstellen, zusammen mit dem absolut chaotischen TI-99/PROTON, ein Computer, bestehend aus einem 40x60 cm Sperrholzbrett mit Lochrasterplatinen und Kilometer Kabeln!

Er wird bis dahin zwar vermutlich noch nicht vollendet sein, aber doch zumindest schon laufen (hoffentlich)!

„TI99/4A-KONSOLE BIS ZU 3MAL SCHNELLER"

In der Anschlußbelegung hat sich leider ein Fehler eingeschlichen. Bitte verwenden Sie nachfolgende Anschlüsse.
Die Anschlüsse A12 der Rambausteine werden nicht mit U504 Pin 1 verbunden, sondern wie folgt:
Ram 1 Pin 2 und Ram 2 Pin 2 werden mit dem 74LS08 Pin 1 verbunden
Ram 3 Pin 2 und Ram 4 Pin 2 werden mit dem 74LS08 Pin 4 verbunden.
Somit ist die Funktion der Schaltung gewährleistet.
Leider laufen aber nicht alle Programme damit, so daß in einem der nächsten Hefte eine softwaremäßige Umschaltung zwischen dem internen und dem externen Speicher veröffentlicht wird.

R. Ziegler

TI99/4A

PREISSENKUNG BEI VIELEN ARTIKELN !!

Peripherie

Computer TI-99/4 A komplett	249,–
Platine TI-99/4 A	59,–
Tastatur TI-99/4 A	39,–
32 K-Karte (Atronic)	249,–
RS 232-Karte (Atronic)	249,–
Disccontroller DSDD (Corcomp)	399.–
P-Code-Karte (gebraucht)	399,–
80-Zeichen-Karte (Mechatronic)	449,–
128 K GRAM-Karte (Mechatronic)	499,–
CPS 99 mit 1 Diskettenlaufwer DSDD + Statistikmodul	999,–
CPS 99 mit 2 Diskettenlaufwerken DSDD + Statistikmodul	1299,–
Diskettenlaufwerk DSDD mit Einbausatz für Peripheriebox	329,–
2 Diskettenlaufwerke DSDD mit Einbau- und Kabelsatz für Peripheriebox	679,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle	249,–
Ext. 128 K-Erweiterung mit Centronicsschnittstelle + Druckerkabel + Epsondrucker LX 800	869,–
dto. + Epsondrucker FX 800	1299,–
dto. + Commodoredrucker MPS 1000	959,–
dto. + Stardrucker NL-10	899,–
RGB-Modulator mit Scartstecker	179,–
RGB-Modulator + Farbmonitor 1081	869,–
MBX-Sprachsteuereinheit + Baseball + Sewermania anschlußfertig	249,–
Grafiktablett Supersketch	99,–
Supersketch + Defender + Dig Dug + Donkey Kong + Statistik	149,–
WIPO Mouse 99 anschlußfertig	179,–
Akustikkoppler Dataphon S 21 d + Kabel für RS 232 Karte	279,–
Modulexpander 3-fach	125,–
Fernbedienung (Original TI)	49,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter)	115,–

Software

Extended Basic (Original TI)	129,–
Extended Basic II Plus	229,–
Mini Memory	149,–
Mini Memory + Assemblerbuch dt.	199,–
Editor/Assembler (32 K + Disc notw.)	149,–
TI-Logo II (32 K + Disc notwendig)	179,–
Diskfixermodul (Navarone)	99,–
Navarone Database monagement	99,–
Terminal Emulator II	59,–
TI-99/4 A Flyer (Flugsimulation)	69,–
Attack, Chisholm Trail, Amazing	je 19,–
Car Wars, Moon Mine, Munch Man, Invaders. Tombstone, Statistik, Defender, Dig Dug	je 29,–
Espial, Fathom, Soccer, Parsec, Treasure Island, Jungle Hunt, Moon Patrol, Shamus, Donkey Kong	je 39,–
Adventuremodul, Buck, Rogers, Burger time, Congo Bongo, Microsurgeon, Moonsweeper, Popeye, Tunnels of doom, Pirates Isle, Pole Position	je 49,–
Video Chess (Schachmodul)	59,–
Dig Dug + Moon Patrol	49,–
Defender + Moon Mine	49,–
Parsec + Shamus	59,–
Espial + Moonsweeper	69,–
Fathom + Burgertime	69,–
Buck Rogers + Jungle Hunt	69,–
Defender + Dig Dug + Jungle Hunt + Shamus	99,–
TI-Artist (Diskette)	99,–
Turbo Pascal (Diskette)	149,–
Window Manager für Turbo Pascal	99,–
C'99 Compiler (Diskette)	99,–

Komplettübersicht über weiteres Zubehör + Modulpakete in unserer Preisliste !!!!!

Bücher

Anwender-Handbuch Turbo Pascal dt.	39,–
Editor/Assembler Handbuch dt.	98,–
TMS 9900 Assemblerhandbuch für das Mini Memory deutsch	78.–
TI-99/4 A Intern dt.	38,–
Assembler-Kurs (99er-Journal)	59,–
Introduction Assembly Language	29,–

Commodore/Schneider

Commodore C 128 D	1049,–
Amiga 500 + Farbmonitor 1081	1769,–
Amiga 2000 + Farbmonitor 1081	3079,–
Commodore Plus 4 + Floppy VC 1551	469,–
Schneider CPC 6128 Grünmonitor	729,–
CPC 6128 Farbmonitor	1149,–
PC 1640/1 Laufwerk/SW-Monitor	1549,–
2 Laufwerke/SW-Monitor	1979,–
1 Laufwerk/Farbmonitor	1979,–
2 Laufwerke/Farbmonitor	2429,–

Versandkostenpauschale (Warenwert bis DM 1000,–/darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–). Lieferung nur gegen NN oder Vorauskasse: Ausland nur Vorauskasse. Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstr. 5, 7324 Rechberghausen, Tel. (0 71 61) 5 28 89

EIN MYSTERIUM AUF DEM TI: DIE CASSETTE

Schon oft wurde in der TI-REVUE das Thema Cassette besprochen. Noch nie aber wurde richtig erklärt, wie man diese aus Assembler aufruft und nutzen kann. Genau dieses soll dieser Bericht nachholen.
Um aber den Cassettenrecorder in Assembler ansprechen zu können, benötigt man einen Assembler. Hierbei ist es gleich, ob es sich um den Editor-Assembler, das Minimem oder einen sonstigen Assembler handelt. Er muß hauptsächlich nur die TMS9900 Assembler-Befehle korrekt in Maschinencode umwandeln. Doch nun zum eigentlichen Problem.

1.) EINLADEN VON PROGRAMM-IMAGE-FILES

Bevor man die Cassette ansprechen kann, muß man zuerst eine sog. „PAB" aufbauen. „BAP" ist die Abkürzung von „Peripheral Access Buffer", der den Speicherbereich darstellt, auf dem die Peripherie, in unserem Fall der Cassettenrecorder, zugreift. Der Aufbau wird anhand des kleinen Beispielprogrammes schnell klar.

```
        DEF   CLOAD,CSAVE,COPEN.CCLOSE
              Startname
        REF   GPLLNK,VMBW,VSBW,VMNR
              Unterprogramme hier E-A-Modul

FAC     EQU   >834A        Für Gerätename
WORD    EQU   >83D0        Wird später >0000
PABNAM  EQU   >8356        Byte hinter Filename
NAMLON  EQU   >8354        Länge des Namens
INDIKA  EQU   >836D        Bei Kassette wird sie >08
STATUS  EQU   >837C        Wird später >00

LOAD    DATA  >0500        >05 -> Wert für Laden
SAVE    DATA  >0600        >06 -> Wert f. Speichern
OPEN    DATA  >0100        >01 -> Wert für Open
CLOSE   DATA  >0200        >02 -> Wert für Close
READ    DATA  >0300        >03 -> Wert für Read
WRITE   DATA  >0400        >04 -> Wert für Write
LONG    DATA   3000        Max. Länge >3000 Bytes
PABADR  DATA   0E00        Adresse f. PAB-Liste i. VDP

*PAB-DATENLISTE
PAB     DATA  >0000        1. Byte: Load, Save, etc.
                           2. Byte: Error
        DATA  >0F00        VDP-Adr. d. gelad. Daten
                           1. Byte: Datensatzlänge
                           2. Byte: Geladene
                                    Character
                           Hier >0000, da Prg.
        DATA  >0003        1. Byte: Screen-Offset
                           2. Byte: Namenslänge
NAMPAB  TEXT  'CS1'        Gerätename
MYWS    BSS   >20          Eigner WS
```

Diese lange Liste von Vorbereitungen sieht viel komplizierter aus, als sie in Wirklichkeit ist. Um eventuelle Unklarheiten zu vermeiden, gehe ich sie am besten Zeile für Zeile durch. Hierbei setze ich die Kenntnis der Assemblerbefehle voraus.

a.) Die Equals

Adresse	Bedeutung
>834A	FAC, an dieser Adresse wird der Name des Gerätes abgelegt, in unserem Fall: 'CS1'
>83D0	Dieses Wort muß auf >0000 gesetzt werden
>8356	Zeigt später auf die erste Adresse hinter dem Name in der PAB im VDP.
>8354	In diesem Wort wird die Namenslänge des Gerätenamens abgelegt, in unserem Fall >0003 für 'CS1'.
>836D	Muß später auf >08 gesetzt werden, um anzuzeigen, daß ein Kassetten-DSR-Aufruf erfolgt.
>837C	Das Statusbyte muß auf >00 gesetzt werden

b.) Die PAB-Datenliste

Byte Nr.	Bedeutung
1	Code für die Art des zu eröffnenden Files >01 -> Open Datei >02 -> Close Datei >03 -> Read Datensatz >04 -> Write Datensatz >05 -> Load Prg.-Image File >06 -> Save Prg.-Image File *>07 -> Delete Datei *>08 -> Scratch Record >09 -> Status
2	Enthält alle Informationen, die das System benötigt. Hierbei gilt: Bit 1–3: Error Code Bit 4: Datensatzlänge; 0=Fixed, 1=Variable Bit 5: Satztyp; 0=Display, 1=Internal Bit 6,7: Operationsmodus: *00=Update, 01=Output, 10=Input, *11=Append Bit 8: Dateityp: 0=Sequential, *1=Relative Dieses Byte wird vom User nicht gesetzt, wenn es sich um ein Prg.-Image File handelt. Es wird in der PAB mit >00 angegeben.

Byte Nr.	Bedeutung
3,4	Die Adresse im VDP, an die die gelesene File geschrieben wird, oder von der das, was gespeichert werden soll, gelesen wird.
5	Datensatzlänge, bei Programm-Image Files setzt man sie auf >00
6	Die Anzahl der Bytes, die bei Write geschrieben werden sollen. Dieses Byte enthält bei Read die Anzahl der gelesenen Bytes.
7,8	Datensatznummer bei Relativen Dateien, da diese bei Kassette aber nicht möglich sind, setzt man es auf >0000. Bei Prog.-Image Format steht hier die Anzahl der max. zu lesenden/der zu schreibenden Bytes.
9	Screenoffset: E/A -> >00 XB -> >60

10 Namenslänge des Gerätenamens.
11ff Gerätename, in unserem Fall 'CS1'.

c.) Die „freien" DATAs

Label	Wert	Bedeutung
LOAD	>0500	Op.-Code f. Load Prg.-Image File
SAVE	>0600	Op.-Code f. Write Prg.-Image File
OPEN	>0100	Op.-Code f. Open Datei
CLOSE	>0200	Op.-Code f. Close Datei
WRITE	>0300	Op.-Code f. Write Datensatz
READ	>0400	Op.-Code f. Read Datensatz
LONG	>3000	Anzahl der zu lesenden/zu schreibenden Bytes
PABADR	>0E00	VDP-Adresse, an die die PAB gelegt wird.

Diese Tabellen sehen um einiges trockener aus, als sie in Wirklichkeit sind. In ihnen ist einfach alles enthalten, was an Information zum ersten Teil des Programmes zu sagen ist.
Wie hoffentlich aufgefallen, sind einige Werte mit Sternchen ('*') gekennzeichnet. Dies sind Werte, die nur mit Diskette zur Verfügung stehen, der Vollständigkeit halber aber genannt werden müssen.
Doch weiter zum Text.
Die Programmvorbereitung sieht vor, die PAB-Daten für mehrere Anwendungen zu verwenden. Die Datenliste allein eröffnet aber noch kein File auf der Kassette. Dazu muß man die PAB-Daten ins VDP legen.
Dies geschieht mit einer einfachen Routine:

```
CLOAD  LWPI  MYWS        eigener WS laden
       LI    R0,PABADR   Adr. der PAB im VDP
       LI    R1,PAB      Adr. der Datenliste
       LI    R2,>000B    Länge der PAB
       BLWP  @VMBW       --> ins VDP
```

Wie ersichtlich, ist in der PAB noch nicht festgelegt, ob eröffnet, geladen, etc. werden soll.
Dazu muß erst noch der betreffende Op.-Code in die PAB im VDP gelegt werden. Wir wollen zunächst ein Programm-Image File laden. Dazu benötigt man auch noch die Anzahl der Bytes, die maximal geladen werden dürfen Die obere Grenze, für die das VDP-RAM noch ausreicht, ist >3000 Bytes. Dies wollen wir hier verwenden.
Die Adresse im VDP für den Load Op.-Code ist PABADR. Die Adresse für die Länge ist PABADR+4.
Das Ablegen der Werte ins VDP geschieht auf dieselbe Weise wie das Ablegen der PAB im VDP.

– Ablegen des Op.-Codes

```
       LI    R0,PABADR     Adr. für den Op.-Code
       LI    R1,LOAD       Op.-Code
       LI    R2,>0002      2 Bytes
       BLWP  @VMBW         --> ins VDP
```

– Ablegen der Länge

```
       LI    R0,PABADR+4   Adr. für Länge
       LI    R1,LONG       Länge
       BLWP  @VMBW         --> ins VDP
```

Ist die PAB im VDP, so tut sich immer noch nicht sehr viel. Der Kassettenrekorder ruht weiterhin sanft auf dem Tisch. Um ihn in Action zu bringen, sind noch einige Kleinigkeiten zu erledigen:
– Es muß in die FAC der Gerätename gelegt werden. In unserem Fall 'CS1'.
– Das Wort auf Adresse >83D0 (Label WORD) muß gelöscht werden.
– Auf Adresse >8356 (Label PABNAM) muß die Adresse des Bytes, das hinter der PAB im VDP steht. In unserem Fall also >0E0B.
– Adresse >836D (Label INDIKA) muß auf >08 gesetzt werden, um einen Kassetten-DSR Call anzuzeigen.
– Das Statusbyte (Adresse >837C, Label STATUS) muß gelöscht werden.
Und schlußendlich
– GPLLNK muß mit der Adresse >003D ausgeführt werden.
Das hört sich nun nach mehr an, als es eigentlich ist. In Assembler sieht das Programm folgendermaßen aus:

```
MOV   @NAMPAB,@FAC        Name--> FAC
MOVB  @NAMPAB+2,@FAC+2
CLR   @WORD               >83D0 löschen
LI    R0,>0E0B
MOV   R0,@PABNAM          Byte hinter PAB
LI    R0,>0800
MOVB  R0,@INDIKA          >836D --> >08
CLR   R0
MOVB  R0,@STATUS          Status löschen

BLWP  @GPLLNK             und ausführen
DATA  >003D
```

Wurde bis hier alles korrekt ausgeführt, so zeigt sich die gewohnte Meldung der Kassettenroutine auf dem Bildschirm. Es wird nun ein Programm-Image File mit der maximalen Länge von >3000 Bytes eingelesen. Diese Daten befinden sich nach dem Laden im VDP ab Adresse >0F00. Von dort aus können sie weiterverarbeitet werden.
Trat ein Lesefehler auf, so zeigt diesen einerseits die interne Laderoutine an. Andererseits steht der Fehlercode auch im zweiten Byte der PAB im VDP. Dort ist es in den Bytes 1–3 enthalten. Von dort aus kann dann der Fehler innerhalb des Programmes weiter verarbeitet werden.
Das wäre eigentlich alles, was über das Einladen von Programm-Image Files zu sagen wäre.

2. ABSPEICHERN VON PROGRAM-IMAGE FILES

Das Abspeichern ist jetzt schnell behandelt. Die Vorbereitungen sind, bis auf ein paar kleine Unterschiede, genau dieselben, wie beim Einlesen.
Zunächst wird *nicht* der Op.-Code für Load, *sondern* der Op-Code für Save verwendet. Es heißt also in dem oben angeführten Assemblerprogramm:

```
       LI    R0,PABADR    Adr. d. Op.-Codes
                          im VDP
       LI    R1,SAVE      Op.-Code
       LI    R2,>0002
       BLWP  @VMBW        ins VDP
```

Die Adresse, an der die Anzahl der maximal einzuladenden Bytes stand, markiert nun die Anzahl der Bytes die geschrieben werden sollen. In unserem Fall werden >3000 Bytes auf die Kassette geschrieben.
Die Adresse, an der die Adresse stand, an die die gelesenen Bytes ins VDP geschrieben werden sollen, markiert nun die Stelle im VDP, von der ab der VDP-Speicherinhalt auf Kassette kopiert werden soll. In unserem Fall werden die Werte an den Adressen V>0F00 bis V>3F00 auf die Kassette geschrieben.
Im Klartext:

– PAB-Byte 7–8 enthält die Anzahl der zu schreibenden Bytes.
– PAB-Byte 3–4 enthält die Startadresse, an der die zu schreibenden Bytes im VDP stehen.
Der Rest ist identisch zum Lesen dieses Filetyps.

3. EINLESEN VON DATEIEN

a.1.) Open Datei "Read"

Vorweg: Wir wollen hier eine Datei im Format sequential/display/variable 80 einlesen, da bei Kassette keine anderen Typen als sequentiell möglich sind. Die Vorbereitungen bis zur Ausführung des DSR-Calls sind dieselben wie bei den Prg.-Image Files. Unterschiede bestehen nur in den Werten, mit denen einige Bytes gesetzt werden.

– Der Op.-Code ist in diesem Fall nun >01 für Open.
– Das zweite Byte wird nun gesetzt, um alle nötigen Informationen über die Datei dem Computer mitzuteilen. Oben wurde schon erwähnt, welche Bedeutung die einzelnen Bits haben. Dies sieht nun folgendermaßen aus:

```
Bit 1–3:  0, wird vom User nicht gesetzt
Bit 4:    1, Variable
          (0 wäre Fixed)
Bit 5:    0, Display.
          (1 wäre Internal)
          (Der Rest ist mit Kassette nicht verfügbar.)
Bit 6,7:  10, Input
          (01 wäre Output)
Bit 8:    0, Sequential
          (1 ist mit Kassette nicht verfügbar.)
```

Der Wert des Bytes ist in unserem Fall also 010100_b, oder >14. Es muß vor dem Ausführen des GPLLNK noch ergänzt werden:

```
      LI     R1,>1200   Wert des Bytes
      LI     R0,>0E01   Adresse im VDP
                        desselben
      BLWP   @VSBW      ins VDP
```

Byte 5 enthält nun auch die maximale Länge der Datensätze, die gelesen werden sollen. In unserem Fall sind es >50 Bytes. Es muß also auf >50 gesetzt werden.

```
      LI     R1,>5000   Wert des Bytes
      LI     R0,>0E04   Adresse im VDP
                        desselben
      BLWP   @VSBW      ins VDP
```

Byte 6 enthält nach dem Lesen eines Datensatzes die Länge in Bytes. Bei Fixed ist dies immer klar, nicht aber bei Variable.
Nun kann das GPLLNK ausgeführt werden.
Der Computer liest aber noch nichts. Er hat lediglich die Datei eröffnet.

a.2.) Open Datei "Write"

Die Vorbereitungen, um auf eine Datei zu schreiben, sind beinahe dieselben. Es gibt nur wenige Unterschiede:
– Byte 2 der PAB hat nun den Wert 00010010_b, also >12. Versucht einmal selbst herauszufinden, was dies bedeutet.
– Byte 5 enthält nun die Anzahl der maximal zu schreibenden Bytes.
– Byte 6 enthält die Anzahl der Bytes, die tatsächlich geschrieben werden. Bei Fixed bleibt die Anzahl gleich, bei Variable variiert diese.
Dies wären die einzigen Unterschiede, die hier zu verbuchen wären. Ob man liest oder schreibt, ist also fast dasselbe.

b.1.) Read Datensatz

Damit er einen Datensatz liest, muß der Op.-Code in der PAB auf >03 gesetzt werden. Der Rest der PAB bleibt unverändert. Das sieht dann folgendermaßen aus:

```
      LI     R0,>0E00   Adresse im VDP
      LI     R1,READ    Op.-Code
      BLWP   @VSBW      ins VDP

      BLWP   @GPLLNK    DSR-Call ausführen
      DATA   >003D
```

Nach dem Lesen stehen die Daten an der in der PAB in Byte 3,4, angegebenen Adresse. Die Anzahl der gelesenen Bytes steht in Byte 6. Ein evtl. aufgetretener Fehler steht in Byte 2, Bit 1–3.
Nun können die Daten weiterverarbeitet werden.

b.2.) Write Datensatz

Auch beim Schreiben der Datensätze handelt es sich um fast dasselbe wie beim Lesen. Lediglich ein paar Bytes haben eine andere Bedeutung.
– Der Op.-Code ist >04 für Write.
Sonst haben alle Bytes dieselbe Bedeutung wie beim Open "Write". Die Sache ist also halb so wild.

c.) Close Datei

Nachdem eine Datei erfolgreich geschrieben, bzw. gelesen wurde, oder nachdem ein Fehler aufgetreten ist, soll man diese ordnungsgemäß schließen. Hierzu genügt es, den Op.-Code zu ändern. Er ist für Close >02. Das anschließende Ausführen der PAB schließt die Datei.
Das war eigentlich alles, was allgemein über die Kassette zu sagen war. Der Rest besteht hauptsächlich aus Probieren und dem Sammeln eigener Erfahrungen. Ich kann dazu nur viel Spaß wünschen.

Oliver Huber

LAUFWERK LIEST NICHT MEHR

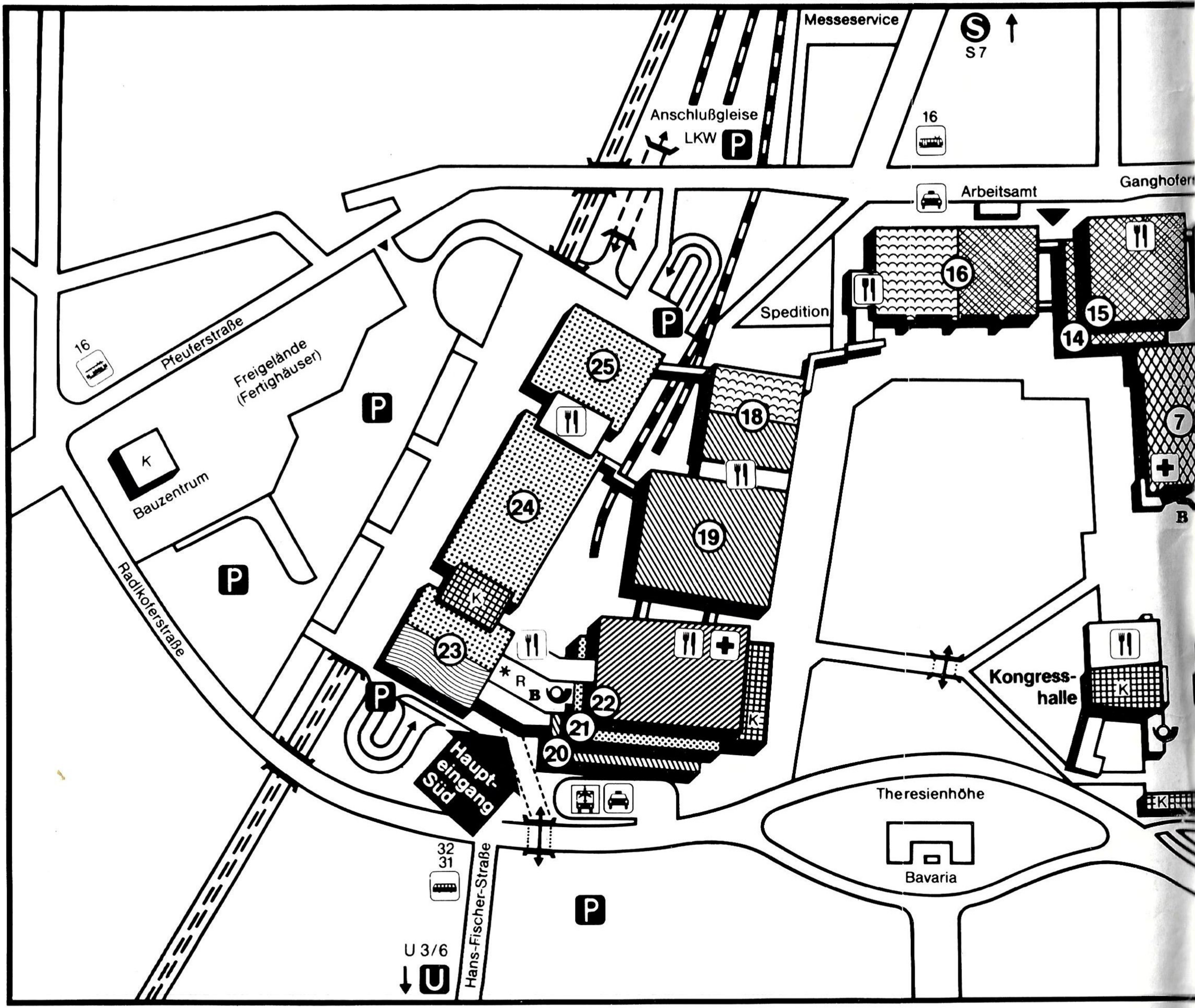
Ich besitze einen TI mit 32k, X-Basic und Diskettenlaufwerk. Seit kurzer Zeit liest jedoch mein Laufwerk nicht mehr richtig. Es fängt plötzlich an, einen Sektor, der existiert, auf der ganzen Diskette zu suchen. Nach einigen Versuchen bricht der Computer mit einer Fehlermeldung ab. Was kann ich tun, damit alles wieder korrekt funktioniert?

Paul Müller, Neu-Ulm

Bei Ihnen ist offensichtlich der Schreib-/Lesekopf des Diskettenlaufwerkes dejustiert. Dies führt dazu, daß der Kopf Sektoren nicht mehr genau „trifft". Er kann sie also nicht mehr erkennen, obwohl sie auf der Diskette existieren. Der einzige Weg dies zu beheben ist, das Laufwerk zum Nachjustieren zu geben. Dies machen viele Fachhändler, bzw. wissen diese, wo man es machen lassen kann.

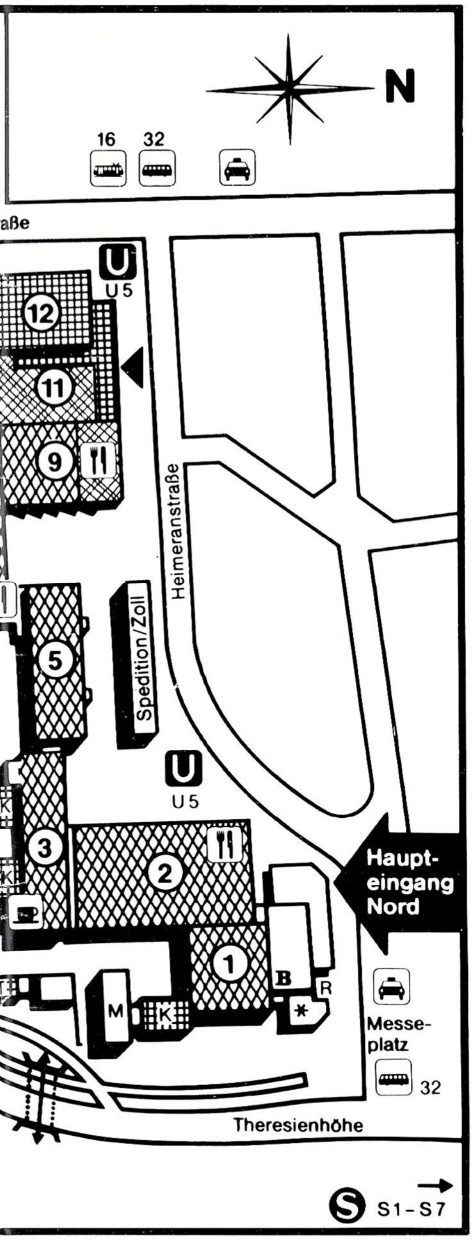
WIR LADEN SIE EIN: SYSTEMS 87-HALLE 21

München, 19.-23. Oktober 1987



M	Messeleitung Fair management Direction du salon	T	Tagungsgebäude Conference building Bâtiment pour réunions	K	Konferenzsäle Conference rooms Salles des conférences	R	Reisebüro Travel agency Agence des voyages	B	B

STAND C10



Software und Dienstleistungen
Beratung — Software — sonstige Dienstleistungen
Software and Services
consultants — software — other services
Software et prestations de services
service conseil — software (logiciel) — autres prestations de services

System-Komponenten (OEM)
Baugruppen — Einzelgeräte
System Components (OEM)
assemblies — individual devices
Composants du système (OEM)
sous-ensembles — appareils individuels

Kommunikationstechnik
Communications Technology
Technique de la communication

Systeme für den Arbeitsplatz Büro
Office Systems
Systèmes concernant le bureau en tant que lieu de travail

Rechenzentrumsausstattungen und Zubehör
Computer-Centre Equipment and Accessories
Equipements pour centres de calcul et accessoires

Informationszentrum
SYSTEMS-Studio — Besucherinformation — Verbände — Verlage und Informationsdienste
Information Centre
SYSTEMS Studio — Visitors' information — Trade associations — Publishing houses and information services
Centre d'information
studio SYSTEMS — information des visiteurs — associations — maisons d'édition et centres d'information

Einstieg in die Informationsverarbeitung
Starting Off in Information Processing
Initiation à l'informatique

Systeme für Wirtschaft und Verwaltung
Business and Administration Systems
Systèmes pour l'industrie et l'administration

k
ique

* Pressezentrum / Press centre / Centre de la presse

Taxi / Taxis

Flughafenbus / Airport shuttle bus / Navette-aéroport

Eingänge / Entrances / Entrées

Es ist geschafft! Obwohl hoffnungslos überbucht – um jeden Quadratmeter gibt es härteste Auseinandersetzungen – konnten wir noch einen Ausstellungsplatz auf der Systems 1987 ergattern. Sie finden uns, wie Sie ja auf unserem Titel bereits gesehen haben, in Halle 21, Stand C 10.

Hiermit laden wir alle unsere Leser, so der Weg sie zur Systems führt, ein, doch einmal vorbeizuschauen.

Allerdings mit einer Einschränkung! Wir bitten um Verständnis, daß auf den uns „im Gnadenwege" zugestandenen 25 Quadratmetern Stand nun wirklich nur ein „geregelter" Besuchsverkehr stattfinden kann. Da, wie Sie ja wissen, in diesem Verlag mehrere Computer-Titel erscheinen, möchte jeder gerne mit der Redaktion „seines" Objektes sprechen können. Das heißt: Wir bitten Sie, sollten Sie Fragen oder Probleme haben, sich an die „Besuchszeiten" zu halten, denn nur dann ist sichergestellt, daß auch die betreffende Redaktion auf dem Stand vertreten ist (die müssen ja auch fürs Heft was tun!). Dementsprechend bitten wir unsere Leser und Besitzer von:

Commodore C16/P4
täglich 9 bis 11
Commodore 64/128
Amiga
täglich 11 bis 13
Schneider (incl. 1512)
täglich 13 bis 15
Atari/MSX
täglich 15 bis 17
und TI
täglich 17 bis 18

Uhr als Gesprächstermin einzuplanen.

TI REVUE

Das Magazin für TI 99-4A

Nr. 9/87

GPL: TEIL 8 UND SCHLUSS

DATEN-TAUSCH: VOM TI ZU JEDEM PC!

DAS HEFT IM HEFT FÜR TI US R

H

S-

EN

TI 99-4A:

DIALOG

FORM-FEED beim TI-WRITER
Eproms kopieren
Probleme mit dem Monitor
File zufällig gelöscht, ist noch was zu retten?
TI-Begeisterung
TI-CLUB Baunatal
ab Seite 40

SERIE & SERVICE

GPL
Dies ist die letzte Folge ab Seite 37

TIPS & TRICKS

INTERNE SPEICHER-ERWEITERUNG
Endlich kompatibel mit der TI-Grafik
Seite 36

VOM TI ZUM PC?
Datenübertragung kein Problem
ab Seite 42

LISTINGS

17 + 4
Eine Abwandlung des bekannten Spieles ab Seite 44

SCRABBLE 86
Wörterlegen auf dem MSX ab Seite 54

DESPERATOR
Dieses Zahlenpuzzle bietet tolle Grafik ab Seite 58

ZWEI-WEG-FREQUENZWEICHE
Ein Programm für Bastler ab Seite 62

Impressum auf Seite 83

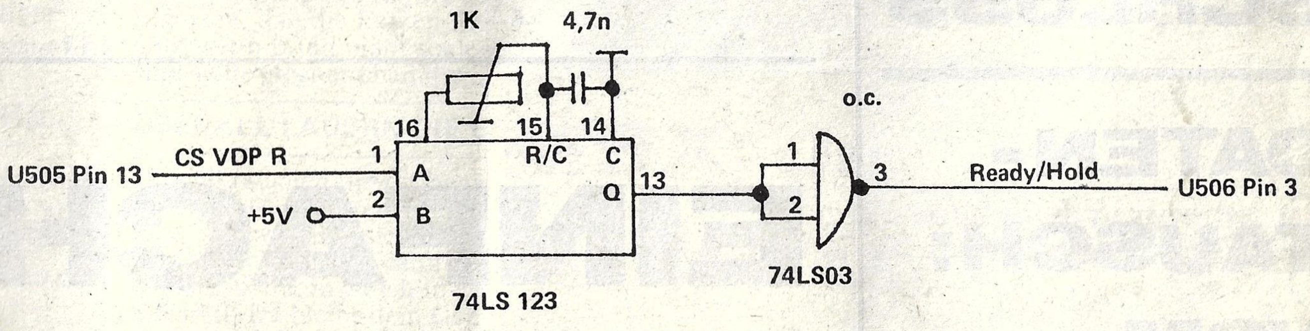
Interne Speicher-Erweiterung endlich kompatibel mit der TI-Grafik

Bisher hat die schnelle 16-Bit-Speichererweiterung speziell bei Grafikausgaben kläglich versagt.
Dieses Manko habe ich nunmehr beseitigt. Nach einigen Vorüberlegungen bin ich zu dem Schluß gekommen, daß das Problem nur bei VDP Read oder Write zu suchen war. Hier erfolgt der Zugriff zu schnell, der VDP-Teil ist nicht mehr in der Lage, so schnell zu reagieren.
Als Lösung bot sich eine Waitstate Logik (Wartezeitgenerierung bei Zugriff auf den VDP-Bereich) an, die in der Testphase möglichst variabel sein sollte. Somit kam der 74LS123 zum Einsatz. In der ersten Version erzeugte die vorliegende Schaltung Waitstates bei VDP Read und VDP Write. Nach genaueren Tests zeigte sich aber, daß die VDP-Daten ruhig schnell geschrieben werden konnten und nur beim Lesen Probleme auftraten.
Somit konnte die Waitstate-Logik auf den Bereich VDP Read limitiert werden.

Wird nun vom VDP Memory gelesen, startet der 74LS123 und liefert ein ca. 1300ns dauerndes Signal (4 Waitstates), welches über einen open collector (o.c.) Treiber (74LS03) in das System eingespeist wird.
Potentiometer-Fetischisten können ihren optimalen Widerstandswert durch tagelange Tests ermitteln, um so auf Nanosekunden zu optimieren. Bei normal Sterblichen genügt der Einbau eines Festwiderstandes von 680 oder 750 Ohm. Es empfiehlt sich, den Wert lieber etwas größer zu wählen.
Mit dieser geringen Modifikation ist die schnelle 32KByte-Speichererweiterung nun voll TI 99/4A-kompatibel.
Eine Anmerkung sei mir noch gestattet: Logischerweise verliert das System jetzt bei Bildschirmzugriffen an Geschwindigkeit, ist aber immer noch schneller als die alte Konsole (Turbo Pascal Compiler ungefähr Faktor 2).

Reiner Ziegler

WaitstateLogik für interne 32K Speichererweiterung



R ≈ 620 Ω C=4700 p

t_{ws}(ns)=0,45·0,62·4700

= 1311,3ns ≈4 Waitstates

©September '87 by R. Ziegler

GPL: Einmal muss Schluß sein. Aber: Es geht gleich weiter!

In diesem abschließenden und somit letzten Teil möchte ich zunächst die letzten beiden GPL-Anweisungen vorstellen, die aber nur für Spezialfälle interessant sind und normalerweise nicht gebraucht werden. Dementsprechend fehlt auch das zugehörige Demoprogramm. Statt dessen gibt es eine allgemeine Betrachtung der Programmiersprache GPL.

Normalerweise werden die GROMs über die allgemein bekannten Adressen 9800 (read data), 9802 (read address), 9C00 (write data) und 9C02 (write address) angesprochen. Zumindest die GROMs der Konsole können Sie aber auch z.B. über die Adressen 9804/9808/9C04/9C08 ansprechen, denn diese sind

SWGR: Wenn mehrere GROMs parallel liegen, wird gewählt

nicht vollständig dekodiert. Im Heft 8/87 präsentierte außerdem Rainer Ziegler seine brillante Idee, mehrere Module gleichzeitig über einen Modulport zu betreiben, der dann selber die genaue Dekodierung bewerkstelligt.
SWGR dient nun dazu, von einem GROM in ein anderes, parallel liegendes, umzuschalten. Der Effekt ist wie ein B bzw. BR oder BS, nur daß eben zusätzlich noch die Basisportadresse im System von >9800 (Standardwert, über den durch Addition die Adressen >9802, >9C00 und >9C02 ermittelt werden) auf einen beliebigen anderen Wert mit geändert wird.

```
Beispiel: DST     §>8300,
                  >980C
          SWGR    §>8300,
                  >6924
```

Nach diesen beiden Anweisungen arbeitet der GPL-Interpreter mit dem GROM der Portadressen 980C/980E/9C0C/9C0E ab der Adresse G6924 weiter.

RTGR: Zurück zum Ursprung

SWGR legt sowohl die augenblickliche Basisportadresse wie auch die GROM-Adresse auf dem Substack. RTGR liest nun diese Daten und geht damit wieder in das alte GROM zurück. SWGR und RTGR verhalten sich somit zueinander wie CALL und RTN.
Kleiner Hinweis für Tüftler: Legt doch einmal ein kleines GPL-Programm ins VDP-RAM und gebt beim SWGR den Wert 8800 als Basisportadresse an. Na, dämmert's? Wichtig dabei aber zu beachten: Jeder VDP-Zugriff und jede Bildschimroperation verändert die VDP-Adresse und wirkt dabei wie ein Sprungbefehl. Bildschirmoperationen sind somit nicht möglich, aber alles andere!

AUSBLICK: WOZU GPL?

Texas Instruments hat lange Zeit ein großes Geheimnis aus dieser Programmiersprache gemacht. Man wollte sich so vor Nachbauten und Kopieren schützen, was bis vor gut zwei Jahren auch ganz gut gelungen war, auch wenn der Erfolg letztendlich die Einstellung des TI war; einen Computer, für den es kaum Software gibt, will eben niemand haben. Mittlerweile ist GPL jedermann zugänglich, sei es mit der GRAM-Karte oder mit Heiner Martins Modulplatine. Aber lohnt es sich überhaupt, sich damit auseinanderzusetzen? Ist GPL mehr als ein inzwischen überholter Kopierschutz? Man sollte sich dazu einmal vor Augen führen, was GPL kann und welche Nachteile es hat:

+ Mit GPL können Sie besser programmieren als mit Maschinencode, da die Anweisungen komplexer sind, d.h., vielfältiger. Allein schon für die ganzen Adreßformate benötigten Sie im Maschinencode jedesmal eine ganze Latte von Subroutinen wie VMBW, VWTR, etc. Im GPL haben Sie diese Probleme nicht. Folge: Ein GPL-Programm ist wesentlich kürzer als ein gleichartiges Maschinenprogramm.
– GPL ist um einiges langsamer als Maschinencode, da GPL ja erst interpretiert werden muß. Man betrachte dazu nur die überwältigende Geschwindigkeit des BASIC-Interpreters!
+ GPL hat viele Befehle, die im Maschinencode ein eigenes Programm benötigen würden. Denken Sie nur an I/O oder FMT. Mit einem FMT können Sie einen gesamten Bildschirm aufbauen, ein I/O übernimmt die vollständige Soundsteuerung.
+ Wenn man erst einmal die ganzen Adreßformate verstanden hat, dann ist GPL einfacher zu erlernen und anzuwenden als Maschinencode. Wer einmal ein längeres Programm in GPL programmiert hat und dann ein ebenso langes im Maschinencode erstellen will, wird sich schnell bewußt, welche Verrenkungen er im Maschinencode für Routinen machen muß, die im GPL manchmal schon in drei Zeilen geschrieben sind.
– GPL ist eine Programmiersprache, die es nur einzig und allein für den TI gibt. Auf anderen Rechnern wäre sie zwar ebenfalls realisierbar, aber angesichts des Angebots höherer Sprachen für diese ist damit wohl nicht zu rechnen.

Diese Liste läßt sich bestimmt noch fortsetzen. Aber man sieht schon: Wer ein umfangreiches Programm erstellen will und dazu viel RAM-Platz für seine Daten benötigt, der sollte in GPL programmieren und das Programm am besten in ein Modul brennen. Auch wem BASIC zu langsam, Maschinencode zu umständlich und höhere Programmiersprachen zu kompliziert oder zu hardwareaufwendig sind (P-Code-Card, Laufwerk, Module . . .), ist mit GPL wirklich nicht schlecht beraten. Für alle anderen Anwendungen hingegen lohnt sich GPL meiner Meinung nach nicht. Der Sinn und Zweck dieser Serie war deshalb auch nicht nur darauf gerichtet, GPL fürs Programmieren neuer Programme und Module darzustellen, sondern vor allem auch fürs Verstehen bereits vorhandener. Sie sollte dem TI wieder ein Stück von seinen mystischen Geheimnissen nehmen, die Texas Instruments ihm gegeben hat.

Sven Dyroff

DIE PROGRAMMIERUNG EINER DSR – TEIL 1

Am Beispiel des MINIMEM möchte ich Ihnen in dieser neuen Serie vorführen, wie man eine DSR für Peripheriegeräte des TI selber programmieren kann. Module wie QSAVE beispielsweise sind nämlich

gar nicht so kompliziert, wie es allgemein aussieht. Als exemplarisches Beispiel für diese Serie benutze ich die in GPL geschriebenen DSRs des MINIMEM, mit dem das Modul bekanntlich unter den Namen MINIMEM, EXPMEM1 und EXPMEM2 als kleine Ramdisk arbeitet, da diese relativ einfach zu verstehen sind.

Vorneweg aber noch ein paar grundsätzliche Dinge zur Programmierung von DSRs, wobei ich zwar die GPL-DSRs zum Vorbild nehme, die Angaben, die sich auf Register und Workspacebereiche beziehen, aber selbstverständlich nur für DSRs im Maschinencode gültig sind.

1. Für DSRs steht der gesamte GROM-Bereich (siehe die entsprechenden GPL-Berichte I und IV) und der ROM-Bereich von >4000 bis >5FFF zur Verfügung, wobei letzterer durch CRU-Anweisungen unter bestimmten Adressen an- und ausschaltbar sein muß, wie schon mehrmals in der TI-Revue und auch anderen Zeitschriften ausführlichst beschrieben.

2. Eine DSR hat nur einen sehr begrenzten RAM-Bereich zum Arbeiten zur Verfügung. Es sind die Bereiche >834A bis >836D und >83DA bis >83DF, wobei jeder DSR diese Bereiche zustehen. Zwischen zwei DSR-Zugriffen können hier also keine Werte erhalten bleiben! Außerdem befindet sich Ihr Workspace im Bereich >83E0 bis >83FF, wobei Sie nur die Register R0 bis R10 benutzen dürfen. R11 enthält Ihre Returnadresse, in R12 steht der CRU-Wert, unter dem der Computer Ihr ROM eingeschaltet hat und es später wieder ausschalten muß, und R13 bis R15 sind die bekannten Systemregister. R13 enthält die GROM-Basisportadresse >9800 R14 Systemflags und R15 die VDP-Basisportadresse >8C02.

3. Vom System bekommen Sie zwei Pointer übermittelt: >8356 zeigt auf den Punkt nach dem Devicenamen im PAB (falls ein Punkt vorhanden, sonst auf das Byte nach dem Namen) und >8354 enthält die Länge des Namens bis zum ersten Punkt. Steht im PAB z.B. 'RS232.BA=300', so weist >8356 auf den Punkt und >8354 enthält den Wert 5. Beide Pointer dürfen von Ihnen verändert werden.

4. Der Rücksprung ins System geschieht mit der Befehlsfolge INCT R11, B *R11. Lassen Sie den INCT nämlich weg, dann sucht der TI weiter, ob noch andere Peripheriegeräte unter diesem Namen vorhanden sind. Das soll er aber normalerweise nicht. Wenn doch, dann darf der Bereich von >834A bis >8357 nicht verändert worden sein, denn >834A bis >8353 enthält eine Kopie des zu suchenden DSR-Namens, und anhand dieser Kopie geht der Suchvorgang vor sich.

5. Errors werden durch Setzen eines Errorcodes im PAB dem System angezeigt. Jeder Code besitzt dabei eine genau definierte Bedeutung, die Sie z.B. im Handbuch zum TI-Basic auf Seite 135f oder, etwas ausführlicher, im Handbuch zum Mini-Memory auf Seite 55 nachschlagen können. Das C-Bit des GPL-Statusbyte wird dabei *nicht* gesetzt!

6. Zugriffe auf das VDP-RAM und auf GROMs sollten nicht direkt, sondern registerindiziert gemacht werden. D.h., statt der Anweisung MOVB R1,§>8800 sollte man besser schreiben MOVB R1,§>FBFE (R15). Der Grund liegt darin, daß sich diese Portadressen ändern können, wie man ja an Beispielen wie der GRAM-Karte und des neuen TI-GENEVE sieht.

So, nun aber zu den eigentlichen

```
G6000 DATA >AA01   GROM-Identifier
G6002 DATA >0000   ???
G6004 DATA >0000   Keine Power-Up-Routine da
G6006 DATA >6010   Pointer für die Programme EASY-BUG
                   und MINIMEM
G6008 DATA >6149   Das ist unser DSR-Pointer!
G600A DATA >6105   Pointer für die CALL-Befehle des
                   Moduls
G600C DATA >0000   Keine Interruptroutine da (in GROMs sowieso
                   nicht möglich!)
G600E DATA >0000   ???
```

Uns interessiert hier nur, was sich hinter dem DSR-Pointer auf G6008 verbirgt:

```
G6149 DATA >6155   Es kommt noch eine DSR
G614B DATA >616D   Die 1. DSR startet bei G616D
G614D BYTE >07     Länge des Namens
G614E TEXT 'MINIMEM'

G6155 DATA >6161   Es folgt noch eine dritte DSR
G6157 DATA >6177   Die 2. DSR startet bei G6177
G6159 BYTE >07
G615A TEXT 'EXOMEM!'

G6161 DATA >0000   Es folgt keine weitere DSR
G6163 DATA >6181   Die 3. DSR startet bei G6181
G6165 BYTE >07
G6166 TEXT 'EXPMEM2'
```

Damit wären wir schon bei den eigentlichen Routinen. Alle drei DSRs haben ja letztendlich die gleiche Aufgabe, unterscheiden sich nur bei den verwendeten Speicherbereichen. Deshalb werden Startadresse und Länge des jeweiligen Bereiches zu Beginn der DSR festgelegt und ab dann von einer einzigen Routine verwaltet:

```
Start MINIMEM
G616D DST   §>835A,>7000   Anfang der 4K des Mini-Memory
G6171 DST   §>835C,>0FF8   Länge des Bereiches minus 8
G6175 BR    GROM§>6189     Weiter

Start EXPMEM1
G6177 DST   §>835A,>2000   Anfang der Low-Memory-Expansion
G617B DST   §>835C,>1FF7   Länge des Bereiches minus 8
G617F BR    GROM§>6189     Weiter

Start EXPMEM2
G6181 DST   §>835A,>A000   Anfang der High-Memory-Expansion
G6185 DST   §>835C,>5FD8   Länge des Bereiches minus 8
```

DSRs und zum ersten Problem: Wie erkennt der TI eine DSR?
Antwort: Dieses Thema wurde im GPL-Bericht I und in einem extra Artikel in der TI-Revue bereits zur Genüge behandelt und ich werde es deshalb hier nicht nocheinmal breittreten. Deshalb gleich ans Eingemachte: Links sehen Sie den Header des Minimem-GROMs:
Nun steht in >835A die jeweilige Startadresse und in >835C die Bereichslänge, von der 8 Bytes als Header benötigt werden, der die Informationen über Art des Files, seine Länge etc. enthält.
Als nächstes werden die ersten 10 Bytes des PAB, die die gesamten Statusinformationen enthalten, in das Scratch-PAD-RAM kopiert. Das ist ein sehr beliebtes Vorgehen, da die Auswertung der Informationen im Scratch-PAD-RAM wesentlich einfacher ist als im VDP-RAM. Beim Programmieren im Maschinencode bekommt man diesen Vorteil noch deutlicher zu spüren:

```
G6189 DST   §>8358,§>8356 Pointer auf Byte nach dem Namen
G618C DSUB  §>8358,§>8354 >8358 zeigt nun auf das erste Namensbyte
                          Namensbyte
G618F MOVE 10 TO §>834A FROM VDP§>FFF6(§>8358)
                          PAB kopieren, wobei >8358 plus
                          >FFF6 auf das erste Byte des PAB zeigt
```

Dann werden die 3 Errorbits im PAB gelöscht:

```
G6197 AND   VDP§>FFF7(§>8358),>1F
```

Nun folgt die Auswertung des ersten Bytes. Es gibt Auskunft darüber, welche Operation überhaupt gewünscht wird (siehe Handbuch zum Ed/Ass):

```
G619D CH    §>834A,>09    Code größer als 9?
G61A0 BS    GROM§>61D8    Ja: Error 03 (Illegal Operation), denn
                              das ist unmöglich!
G61A2 CASE  §>834A        Routine je nach OP-Code
G61A4 BR    GROM§ >61E1   Code 0: OPEN
G61A6 BR    GROM§ >6251   Code 1: CLOSE
G61A8 BR    GROM§ >6253   Code 2: READ
G61AABR     GROM§ >62C2   Code 3: WRITE
G61ACBR     GROM§ >634E   Code 4: RESTORE
G61AEBR     GROM§ >6365   Code 5: OLD bzw. LOAD
G61B0 BR    GROM§ >6395   Code 6: SAVE
G61B2 BR    GROM§ >6433   Code 7: DELETE
G61B4 BR    GROM§ >61D8   Code 8: SCRATCH RECORD
G61B6 BR    GROM§ >63C3   Code 9: EOF
```

Wie Sie sehen, ergibt auch die Anweisung SCRATCH RECORD einen Error, denn das Mini-Memory kennt diese Anweisung nicht. Wo wir schon bei den Errors sind, sie stehen im GROM gleich im Anschluß an die obige Sprungtabelle. Das Mini-Memory kennt fünf verschiedene Errors, die es einfach dadurch mitteilt, daß es dafür einen festgelegten Errorcode (siehe Handbuch zum Ed/Ass oder zum Mini-Mem, S. 55, oder zum TI-Basic, S. 135f) in den PAB schreibt und dann ins Basic zurückkehrt:

```
Error 02: Bad Attribute
G61B8 OR VDP§>FFF7(§>8358),>40
G61BE BR GROM§>61DE

Error 04: Buffer full
G61C0 OR VDP§>FFF7(§>8358),>80
G61C6 BR GROM§>61DE

Error 05: Read past EOF
G61C8 OR VDEP§>FFF7(§>8358),>A0
G61CE BR GROM§>61DE

Error 07: File Error
G61D0 OR VDP§>FFF7(§>8358),>E0
G61D6 BR GROM§>61DE

Error 03: Illegal Operation
G61D8 OR VDP§>FFF7(§>8358),>60

GSR-Return to Basic
G61DECALL GROM§>0012
```

Das war's auch schon für heute. Die Routinen für die einzelnen Operationen werde ich dann in den kommenden Folgen vorstellen.
Hoffentlich dient das als Anregung für die Programmierer unter Ihnen, eigene Peripheriegeräte zu entwickeln. Zumindest haben Sie dann keine Schwierigkeiten mehr, die Software dazu zu erstellen.

Sven Dyroff

ZEITSCHLEIFE FÜR XB

Oft werden bei Programmen, die auf bestimmte Peripherie zugreifen, präzise Warteschleifen benötigt. Diese Routine eignet sich dafür sehr gut, da sie sehr genau arbeitet. Erzielt wird die Genauigkeit dadurch, daß in der Zeitschleife Befehle verwendet werden, deren Ausführung genau 1 sec.*SEC dauert.
Das Programm wird mit CALL LOAD ... geladen und mit CALL LINK ("WAIT",SEC) gestartet. Der Parameter SEC darf dabei nicht größer als dez. 65536 werden.

```
********************
* Zeitschleife für XB  *
* Copyright by         *
* Rodolfo Bordenave    *
* Argentinien          *
********************

        DEF WAIT
MYWS    BSS 32
NUMREF  EQU >200C
XMLLNK  EQU >2018
WAIT    LWPI MYWS
        CLR R0
        LI R1,1
        BLWP §NUMREF
        BLWP §XMLLNK
        DATA >12B8
        MOV §>834A,R0
        JEQ RET
FOR     LI R2,2
B       LI R1,>A2B6
A       DEC R1
        JNE A
        DEC R2
        JNE B
NEXT    DEC R0
        JNE FOR

RET     CLR §>837C
        LWPI >83E0
        B §>0070
        END
```

NOCHMALS: FORM-FEED BEIM TI-WRITER

Unter der Rubrik Tips & Tricks erschien in TI-REVUE 4/87 eine Leserzuschrift über das Thema „Form-Feed beim TI-Writer ausschalten“. Hierzu habe ich folgenden Hinweis: Mit Hilfe eines Basic-Programms läßt sich das lästige Form-Feed bei einem Epson LX-800 auf eine Zeile reduzieren. Dies funktioniert sicherlich auch bei anderen Epson-Druckern. Es sind die Steuerzeichen CHRS(27);“C“;CHRS(1).
Werner Stresemann, Berlin

Vielen Dank für Ihren Tip. Noch kurz etwas zu der Steuersequenz. ESC 6 7 legt beim Epson LX-800 die Größe des Form-Feeds in Zeilen fest, man kann also bestimmen, wieviele Zeilen beim Form-Feed ausgelassen werden sollen. Diese Sequenz existiert auf den meisten Druckern, so daß es nur eine Modulationsfrage ist. Ihr Trick vermag also das Form-Feed nicht zu unterdrücken, sondern vernachlässigbar klein zu machen.

EPROMS KOPIEREN

Ich habe den in Ihrer Zeitschrift veröffentlichten Eprommer nachgebaut und bin sehr zufrieden damit. Ich habe jedoch folgendes Problem: Ich besitze ein Eprom, dessen Programm ich gerne in ein anderes Eprom kopieren möchte. Leider sind meine Assemblerkenntnisse nicht so gut, daß ich dafür ein Programm schreiben kann. Existiert schon ein solches Programm, das den Inhalt eines Eproms auf Diskette speichert und anschließend auf ein anderes Eprom diesen Inhalt überträgt? Ich wäre dafür sehr dankbar.
Michael Dockendorff, Bad Kreuznach

Ein Programm, das den Eprom-Inhalt auf Diskette speichert, existiert unseres Wissens nach nicht. Sie können aber mit der Eprommersoftware ein Eprom kopieren. Stecken Sie das Eprom, das gelesen werden soll, in den Eprommer und lesen Sie es aus. Dann stecken Sie das Eprom ein, das geschrieben werden soll und programmieren es. Dabei dürfen Sie jedoch nicht die vorher eingegebenen Parameter ändern. Ein weiterer Punkt, auf den Sie achten müssen, sind die Urheber-Rechte. Copyright-Verletzungen sind strafbar und werden polizeilich verfolgt.

NOCHMAL MONITOR-PROBLEME

Ich versuche seit einiger Zeit, meinen Computer an einen Farmonitor anzuschließen. Die Eingänge am Monitor sind drei Cinch-Buchsen für Videosignal, Farbsättigung und Ton. Durch den Anschluß eines 6poligen Diodensteckers am TI-Ausgang habe ich Ton und ein s/w-Bild erhalten, jedoch keine Farbe. Den Stecker habe ich folgendermaßen belegt:
2 = Videosignal
4 = Farbsättigung
5 = Ton
6 = Masse
Besteht die Möglichkeit, dem Monitor doch noch Farbe zu entlocken?
J. W. München

In der Hinsicht auf Farbe muß ich Sie leider enttäuschen. Der Eingang des Monitors entspricht einer PC-Norm, die mit dem TI99 nicht zu erreichen ist. Es besteht also keine Möglichkeit, dem Monitor Farbe zu entlocken.

FILE ZUFÄLLIG GELÖSCHT

Durch einen Eingabefehler wurde auf einer meiner Disketten eine Datei durch ‘DELETE‘ gelöscht. Gibt es eine Möglichkeit, diese noch zu retten, oder ist sie hoffnungslos verloren?
Karl Müller Karlsruhe

Es besteht eine Möglichkeit, diese Datei wieder zu retten. Diese ist aber sehr aufwendig. Zunächst suchen Sie sich die Sektoren heraus, in denen die Datei gespeichert ist. Diese Sektoren wurden durch das DELETE in der Sektor-Belegungstabelle wieder als frei eingetragen. Sie müssen also wieder auf belegt gesetzt werden. Diese Tabelle ist in Sektor 0. Dann müssen Sie im Diskettenkatalog einen neuen Eintrag erstellen, mit Dateiname, Länge, etc. Dieser Eintrag muß dann ordnungsgemäß in die ‘Katalogliste‘ in Sektor 1 eingefügt werden. Wurde alles korrekt ausge-

TI99/4A

PREISSENKUNG BEI VIELEN ARTIKELN !!

Peripherie

Computer TI-99/4 A komplett	249,–
Platine TI-99/4 A	59,–
Tastatur TI-99/4 A	39,–
32 K-Karte (Atronic)	249,–
RS 232-Karte (Atronic)	249,–
Disccontroller DSDD (Corcomp)	399.–
P-Code-Karte (gebraucht)	399,–
80-Zeichen-Karte (Mechatronic)	449,–
128 K GRAM-Karte (Mechatronic)	499,–
CPS 99 mit 1 Diskettenlaufwer DSDD + Statistikmodul	999,–
CPS 99 mit 2 Diskettenlaufwerken DSDD + Statistikmodul	1299,–
Diskettenlaufwerk DSDD mit Einbausatz für Peripheriebox	329,–
2 Diskettenlaufwerke DSDD mit Einbau- und Kabelsatz für Peripheriebox	679,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle	249,–
Ext. 128 K-Erweiterung mit Centronicsschnittstelle + Druckerkabel + Epsondrucker LX 800	869,–
dto. + Epsondrucker FX 800	1299,–
dto. + Commodoredrucker MPS 1000	959,–
dto. + Stardrucker NL-10	899,–
RGB-Modulator mit Scartstecker	179,–
RGB-Modulator + Farbmonitor 1081	869,–
MBX-Sprachsteuereinheit + Baseball + Sewermania anschlußfertig	249,–
Grafiktablett Supersketch	99,–
Supersketch + Defender + Dig Dug + Donkey Kong + Statistik	149,–
WIPO Mouse 99 anschlußfertig	179,–
Akustikkoppler Dataphon S 21 d + Kabel für RS 232 Karte	279,–
Modulexpander 3-fach	125,–
Fernbedienung (Original TI)	49,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter)	115,–

Software

Extended Basic (Original TI)	129,–
Extended Basic II Plus	229,–
Mini Memory	149,–
Mini Memory + Assemblerbuch dt.	199,–
Editor/Assembler (32 K + Disc notw.)	149,–
TI-Logo II (32 K + Disc notwendig)	179,–
Diskfixermodul (Navarone)	99,–
Navarone Database monagement	99,–
Terminal Emulator II	59,–
TI-99/4 A Flyer (Flugsimulation)	69,–
Attack, Chisholm Trail, Amazing	je 19,–
Car Wars, Moon Mine, Munch Man, Invaders. Tombstone, Statistik, Defender, Dig Dug	je 29,–
Espial, Fathom, Soccer, Parsec, Treasure Island, Jungle Hunt, Moon Patrol, Shamus, Donkey Kong	je 39,–
Adventuremodul, Buck, Rogers, Burger time, Congo Bongo, Microsurgeon, Moonsweeper, Popeye, Tunnels of doom, Pirates Isle, Pole Position	je 49,–
Video Chess (Schachmodul)	59,–
Dig Dug + Moon Patrol	49,–
Defender + Moon Mine	49,–
Parsec + Shamus	59,–
Espial + Moonsweeper	69,–
Fathom + Burgertime	69,–
Buck Rogers + Jungle Hunt	69,–
Defender + Dig Dug + Jungle Hunt + Shamus	99,–
TI-Artist (Diskette)	99,–
Turbo Pascal (Diskette)	149,–
Window Manager für Turbo Pascal	99,–
C'99 Compiler (Diskette)	99,–

Komplettübersicht über weiteres Zubehör + Modulpakete in unserer Preisliste !!!!!

Bücher

Anwender-Handbuch Turbo Pascal dt.	39,–
Editor/Assembler Handbuch dt.	98,–
TMS 9900 Assemblerhandbuch für das Mini Memory deutsch	78.–
TI-99/4 A Intern dt.	38,–
Assembler-Kurs (99er-Journal)	59,–
Introduction Assembly Language	29,–

Commodore/Schneider

Commodore C 128 D	1049,–
Amiga 500 + Farbmonitor 1081	1769,–
Amiga 2000 + Farbmonitor 1081	3079,–
Commodore Plus 4 + Floppy VC 1551	469,–
Schneider CPC 6128 Grünmonitor	729,–
CPC 6128 Farbmonitor	1149,–
PC 1640/1 Laufwerk/SW-Monitor	1549,–
2 Laufwerke/SW-Monitor	1979,–
1 Laufwerk/Farbmonitor	1979,–
2 Laufwerke/Farbmonitor	2429,–

Versandkostenpauschale (Warenwert bis DM 1000,–/darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–). Lieferung nur gegen NN oder Vorauskasse: Ausland nur Vorauskasse. Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstr. 5, 7324 Rechberghausen, Tel. (0 71 61) 5 28 89

führt, kann die Datei wieder gelesen werden.

TI-BEGEISTERT

Bisher wurde noch nie ein Lob auf den guten, alten TI ausgesprochen. Es ist doch immerhin eine enorme Leistung, daß sich ein Kleincomputer so viele Jahre auf dem Markt hält, wie der TI. Obwohl die Marken sich abwechseln wie das Wetter, existiert immer noch eine eingeschworene Gruppe von TI-Usern, die sich ihren inzwischen alten Bekannten nicht nehmen läßt. Dies zeugt für mich von einer gewissen Überlegenheit gegenüber den anderen Computern, die meinen, mehr zu können.
Xaver Aichinger
Stuttgart

TI-CLUB BAUNATAL – DER CLUB MIT DEM GUTEN SERVICE!

Nach längerer Pause haben sich nun wieder Clubs gefunden, die in der TI-REVUE informieren wollen. Diesmal sind es zwei an der Zahl. Den Anfang macht der TI-Club Baunatal. Er ist ein „alter Bekannter" in der TI-REVUE.

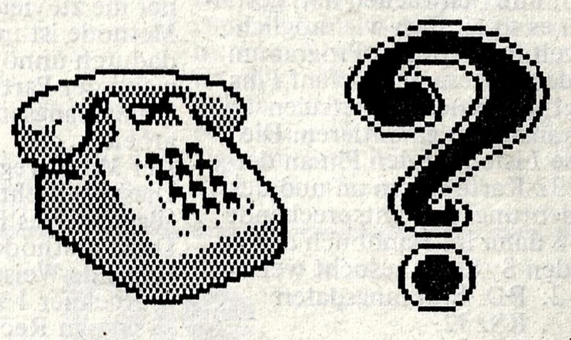
Schon seit Anfang Januar 1985 existiert der TI-Club Baunatal! Niemand hätte damals geahnt, daß aus dem kleinen TI-Club mal ein großer TI-CLUB BAUNATAL mit mittlerweile über 190 Mitgliedern werden würde. Genau wie unsere Mitgliederzahl stieg auch das Angebot, welches wir für unsere Mitglieder haben: Angefangen bei unserem Programmspeicher über die DFÜ-Leitung (Software per Akustikkoppler zu bekommen) bis hin zu unserer TCB-Bestellserviceliste, in der alle Mitglieder reichlich Hardware, und auch Software, finden. So können wir derzeit weit über 80 Module, viele Bücher und auch andere Hardware für den TI günstig anbieten. Desweiteren erhalten wir von TI-Händlern gute Rabatte, so daß wir neue Hardware für das texanische Wunderkind gerne an unsere Mitglieder weitergeben, die diese natürlich kräftig ausschöpfen!!! Zu unserem weiteren Angebot gehören unser beliebter Resetknopf, Kabel für RS232- und Centronicsschnittstellen sowie Assemblersoftware, die wir auf Cassette in Verbindung mit der 32K anbieten können. Wer Informationen rund um den TI haben möchte, ist mit der TI-Post, unserem Cluborgan, bestens bedient. Hier werden aktuelle Angebote unterbreitet, hier gibt es Vorstellung neuer und neuester TI-Hard- und Software und hier finden auch Leserbriefe und -meinungen ihren Platz. Zur Zeit läuft übrigens ein Lernkurs für die Programmiersprache Turbo-Pascal, der von unserem Pascalexperten Dietmar Lucas geschrieben wird und besten Anklang findet. Wer mehr Informationen über uns, unsere Arbeit und unsere Angebote haben möchte, fordere gegen 50 Pfennig Rückporto ein Info an, oder bestelle gegen 3,– DM unsere aktuelle Ausgabe der TI-POST mit Info. Telefonische Anfragen werden unter folgenden Rufnummern gerne entgegengenommen: Herrn Orf erreichen Sie bis 21.30 Uhr täglich unter 0561/491 1078. Herr Göbel ist unter 05604/6999 oder 05602/20 59 erreichbar. Wir hoffen, Ihr Interesse geweckt zu haben und freuen uns auf viel Post von nicht Schreibfaulen TI-Usern, wovon es ja wohl noch einen ganzen Batzen gibt, oder...???
TI-CLUB BAUNATAL
Matthias Orf
Birkenallee 34
3507 Baunatal 1
Tel. 0561/4911078

NEUES VOM TEXAS INSTRUMENTS CLUB KIEL (TICK)

Der zweite Club wurde ebenfalls schon in der TI-REVUE vorgestellt, ist aber noch nicht so bekannt. Es handelt sich hierbei um den TI-Club Kiel. Bei diesem Club ergab sich eine Änderung bezüglich des Service. Es existiert nun eine Clubzeitschrift.

Aufgrund einiger neuer Überlegungen muß die letzte Vorstellung unseres Clubs in folgender Form ergänzt bzw. berichtigt werden: Eine Clubzeitung ist in Arbeit! Die erste Ausgabe wird pünktlich zum TI-Treffen in Bremen erscheinen. Da sich daraus einige Kosten ergeben, kommen wir leider nicht mehr um einen Club-Beitrag herum. Wir glauben aber, daß ein Beitrag von DM 3,– pro Monat nicht zu hoch gegriffen ist. Die Zeitung wird voraussichtlich alle 1 bis 2 Monate erscheinen und einen Umfang von ca. 10 bis 20 DIN-A4-Seiten haben. Geplant sind die festen Rubriken Adventures, Spielbeschreibungen, Kleinanzeigen und Fragen rund um den TI. Die Beschränkung auf den Raum Schleswig-Holstein ist nicht mehr wirksam! Da wir auch schon Anfragen aus anderen Bundesländern erhalten haben, kann ab jetzt jeder TI-User, der Interesse an unserem Club hat, diesem beitreten. Weiterhin bestehen bleibt allerdings die Bitte, bei Anfragen an uns das Porto für die Antwort nicht zu vergessen. Wer sich genauer informieren will, der kann sich an folgende Adresse wenden:

Texas Instruments Club Kiel
c/o Michael Rittweger
Nissenstraße 12
D-2300 Kiel 14
Tel.: (0431) 725566



TI-REVUE Hotline
Dienstags
15.00 bis 19.00
Tel.: 089/129 80 11

Vom TI zum PC? Datenübertragung kein Problem

Immer mehr TI-User kaufen sich neue Systeme und steigen um. Diese traurige Tatsache kann man leider nicht von der Hand weisen. Dieser Bericht soll nun denjenigen gewidmet sein, die ihren TI gegen einen neuen Rechner umtauschen. Er soll zwischen dem TI und dem „Neuen" eine Brücke bilden. Als ich mir einen neuen Rechner zulegte, stand ich vor dem wahrscheinlich normalen Problem: „Wie bekomme ich meine Unsummen von Daten, Texten, Adressen etc. in den Speicher des neuen Rechners?" Sofort kam mir die RS232 in den Sinn. Doch wie soll ein Programm gestaltet sein, das mit einem Minimum an Aufwand ein Maximum an Komfort bietet? Um dieses Nachdenken den anderen zu ersparen, dieser Bericht.

Das wichtigste an allem ist die RS-232-Schnittstelle. Ohne eine solche geht es nicht. Es wird also sowohl am TI als auch an dem neuen System eine solche vorausgesetzt. Man muß also nur noch ein entsprechendes Kabel herstellen. Da jede RS232-Schnittstelle, sei es Atari, Commodore oder

DIE HARDWARE

auch TI, ihre Schwächen hat, gestaltet man es so einfach wie möglich und regelt den Rest im Programm. Es werden insgesamt nur fünf Pins benötigt, um einen fehlerfreien Datentransfer zu garantieren. Die folgende Liste gibt den Pin an der TI-RS232-Karte intern an und dessen Bedeutung. Der entsprechende Pin muß dann im Handbuch des betreffenden Systems gesucht werden.

1.) Pin 2: RD, Empfangsdaten RS232.
2.) Pin 3: TX, Sendedaten.
3.) Pin 5: CTS, Sendebereitschaft (nicht unbedingt erforderlich).
4.) Pin 7: Erde
5.) Pin 20: DTR, Bereitschaft der Datenendeinrichtung (nicht unbedingt erforderlich).

Im Endeffekt werden nur Pin 2, 3 und 7 benötigt. Es reicht völlig aus, ein Kabel mit diesen drei Pins zu fertigen. Der folgende Bericht geht von einem solchen vereinfachten Kabel aus.

DAS ÜBERTRAGEN SEQUENTIELLER DATEIEN

Das Übertragen sequentieller Dateien ist an sich die einfachste Sache, die man sich wünschen kann. Man muß nur mehr oder weniger auf Kleinigkeiten achten, damit alles reibungsfrei funktioniert.

Nun kann man alles nacheinander in den benachbarten Rechner pumpen, ohne Pause, ohne Abfrage. Diese Methode ist abzulehnen, da man, weil der Partner nicht empfangsbereit ist, Daten verlieren kann.

Man kann auch eine feste Pause einbauen, die dafür sorgt, daß der Partner nie zu viele Daten erhält. Diese Methode ist unpraktisch, da man dadurch unnötig Zeit verbraucht, wenn der Partner einmal schneller, einmal langsamer die Daten verarbeitet.

Der Mittelweg ist also angesagt. Dieser besteht darin, daß man ein gegenseitiges Handshake einbaut. Diese Methode funktioniert auf folgende Weise:

– Rechner 1 sendet einen Datensatz zu Rechner 2.
– Während Rechner 2 diesen verarbeitet, wartet Rechner 1.
– Rechner 2 sendet ein 'Ok' zu Rechner 1 und teilt ihm so mit, daß er wieder empfangsbereit ist.

In Extended Basic sieht dies folgendermaßen aus:

```
10!
20!hier Programmanfang
30!
40!a$ ist Datensatz
50!
60Print #1:A$
70Input #1,OK$
80Goto 60
eventuell: 75 For I=1 to 2 :: Next I
```

Mit 'Print #1:A$' wird der zu übertragende Datensatz übertragen. Bei 'Input #1,OK$' wartet der Rechner so lange, bis ein Byte in der RS232 ankommt. Dieses Warten ist in der Software der RS232 integriert. Kam das 'Ok' an, sendet er den nächsten Datensatz.

Ist der Partner ein nicht sehr schneller Rechner, so muß vor dem nächsten Senden noch eine minimal kurze Warteschleife eingebaut werden, um dem Partner Zeit zu lassen, die Read-Routine zu starten. In der Regel genügt dann ein Zählen auf 2 oder 5.

Das war schon der ganze Zauber, der in einem Übertragungsprogramm für sequentielle Dateien steckt. In der Praxis sieht das folgendermaßen aus: Nehmen wir an, man will die Datei 'TEST' vom TI auf einen Atari 1040STF übertragen. Die verwendeten Programmiersprachen sind Extended Basic und GfA-Basic.

Das Programm am TI sieht folgendermaßen aus:

```
10 OPEN #1,"DSK1.TEST",
   INTERNAL,INPUT,FIXED 80
20 OPEN #2,"RS232.BA=9600.
   DA=8.PA=N"
30 LINPUT #1:A$
40 PRINT #2:A$
50 INPUT #2:OK$
60 IF EOF(1)=0 THEN 30
70 PRINT #2"!! EOF !!"
80 CLOSE #2
90 CLOSE #1
100END
```

Das Dateiformat: Was bei anderen Rechnern als ASCII-Format bezeichnet wird, ist beim TI das Format 'DISPLAY'. 'INTERNAL' ist ein TI-Internes Format, das kein anderer Rechner, außer dem TI, besitzt. Es wird 'LINPUT' verwendet, da in Texten in der Regel auch Kommata vorkommen.

Das Programm am Atari 1040STF:

```
Open "U",#1,"AUX:"
Open "O",#2,"Test"
Do
   Line Input #1,A$
   Exit If A$="!! EOF !!"
   Print #2,a$
   Print #1,"OK"
Loop
Close #1
Close #2
```

Nun wird auch Zeile 70 des TI-Programms klar. Sie zeigt dem Partner, daß das Dateiende erreicht ist.

Das GfA-Basic-Programm bildet so genau das Gegenstück zu dem TI-Programm und ermöglicht das Übertragen sequentieller Dateien.

Es sind nur geringfügige Modulationen notwendig, um das Programm auf jedem Rechner, der mit einer RS-232 ausgestattet ist, laufen zu lassen.

Achtung!!! Es ist unbedingt darauf zu achten, daß die Einstellung der RS232 identisch mit der der Gegenstation ist. Eventuelle Unterschiede haben zur Folge, daß beide Compu-

ter aneinander vorbeikommunizieren. Der Erfolg läßt dann ewig auf sich warten, da beide Rechner nicht verstehen, was der andere sagt.
Um eindeutige Klarheit über den Ablauf des Programms und dessen Aufbau zu geben, noch einmal den Aufbauplan.

- Eröffnen beider RS232 (Quelle –>Schreiben, Senke–>Lesen)
- Senden eines Datensatzes.
- Die Quelle wartet, bis die Senke ihr OK zurückmeldet.
- Die Quelle verarbeitet den Datensatz und gibt, sobald sie wieder empfangsbereit ist, ein OK zurück.
- Weiter senden . . .
- Bei EOF schließt man alle Dateien, *auch die der Senke!!*

Hält man sich einigermaßen an dieses Rezept, so wird es kaum Probleme bereiten, eine lange, sequentielle Datei zu senden und während dessen einen Kaffee trinken zu gehen. Doch leider besteht die Welt nicht nur aus sequentiellen Dateien. Es existiert noch ein weiteres Problemfeld:

ÜBERTRAGUNG RELATIVER DATEIEN

Eine relative Datei zu übertragen, erfordert einen geringen Aufwand an Beschaffung von Vorinformation. Eine direkte Übertragung ist nicht möglich. Es muß der Umweg über die sequentielle Datei gemacht werden. Doch alles der Reihe nach.
Zunächst die benötigten Vorinformationen:

- Wie ist der einzelne Datensatz der relativen Datei aufgebaut (Inhalt des Satzes)?
- Kann die Senke relative Dateien verarbeiten?
- Ja: Wie geht das?
- Nein: Welches sequentielle Format wähle ich statt dessen?

Das ist schon alles. Betrachten wir einfach einmal das Programm von TI-Seite aus. Es soll eine relative Datei mit 100 Datensätzen (von 1–100) auf einen Atari 1040STF übertragen werden. Ein Datensatz enthält Adressen und besteht aus 4 Teilstrings. Wie sieht das Programm am TI aus?
Die Anzahl der Datensätze muß nicht unbedingt bekannt sein, sie vereinfacht aber das Programm ein wenig.

```
10 OPEN #1,"RS232.BA=9600.
   DA=8.PA=N"
20 OPEN #2,"DSK1.ADRESSEN",
   INTERNAL,VARIABLE80,
   RELATIVE,INPUT
30 FOR I=1 TO 100
40 INPUT #2:REC I,A$,B$,C$,D$
50 PRINT #1:A$
60 INPUT #1:O$
70 PRINT #1:B$
80 INPUT #1:O$
90 PRINT #1:C$
100INPUT #1:O$
110PRINT #1:D$
120NEXT I
130CLOSE #1
140CLOSE #2
```

Das Programm läßt sich noch vereinfachen, ist aber so übersichtlich und sicherer in der Übertragung. Durch das Bekanntsein der Anzahl der Datensätze erspart man sich eine Abfrage auf EOF. Nach jedem gesendeten String wird auf ein OK gewartet. Dies ist nicht immer notwendig, aber sicherer. Bei der Datenübertragung sollte weniger auf die Geschwindigkeit geachtet werden als auf die Sicherheit.
Zum Programmaufbau muß nichts mehr gesagt werden. Er ist hoffentlich in dieser Form klar. Doch nun das Gegenstück auf dem Atari:

```
Open "U",#1,"AUX:"
Open "R",#2,"Adressen",Len 255
Field #2,A1$ As 255,A2$ As 255,
      A3$ As 255,A4$ As 255
For  I=1 To 100
      Line Input #1,A1$
      Print #1,"OK"
      Line Input #1,A2$
      Print #1,"OK"
      Line Input #1,A3$
      Print #1,"OK"
      Line Input #1,A4$
      Put #2,I
      Print #1,"OK"
Next I
Close #1
Close #2
```

Es ist wichtig, darauf zu achten, daß man das vom Zielcomputer vorgeschriebene Format der Datei einhält.
Besitzt dieser jedoch keine Möglichkeit zur Eröffnung einer relativen Datei, so muß man wohl über übel ein sequentielles Format wählen.
Es empfiehlt sich hierbei, die Datei folgendermaßen aufzubauen:

- Jede Teilinformation eines relativen Datensatzes wird zu einem eigenen Datensatz.
- Die Reihenfolge der Anordnung der Informationen darf nicht wechseln.
- Will man einen Datensatz einlesen, so liest man einfach so lange Hilfsvariablen ein, bis man am Beginn des entsprechenden Satzes ist.
- Der Datensatz wird als „Sammlung" von mehreren Datensätzen eingeladen.

Hierbei empfiehlt es sich, einen Datensatz an den Kopf der Datei zu stellen, der die wichtigsten Informationen enthält:

- Anzahl der Datensätze.
- Wieviele Sätze bilden eine Information?
- Evtl. Anzahl der Informationen.

Diese kann man aber auch mit dem Quotienten von Datensatzanzahl berechnen und der Anzahl der Sätze, die eine Information bilden.

Mit dieser Anleitung muß es jetzt eigentlich ganz einfach möglich sein, den guten alten TI mit einem anderen Rechner via Kabel kommunizieren zu lassen. Der Komfort, den nun das Übertragungsprogramm bietet, liegt allein in der Hand des Programmierers. Der Phantasie sind dort keine Grenzen gesetzt. Die einzigen technischen Vorkenntnisse, die notwendig sind, liefert der Text.

Oliver Huber

Die TI-REVUE soll weiter-leben! Wir suchen noch gute Autoren!

17+4 - Einmal anders

Zu diesem Spiel benötigt man die TI99/4A-Konsole, ein Disk-Laufwerk, das Extended-Basic-Modul, den Speech-Synthesizer und Joystick(1). Eine 32k-Erweiterung ist nicht notwendig!
Dieses Spiel ist eine Abwandlung des bekannten 17 + 4.
Ziel dieser Variation ist es, als Mannschaft oder bei nur einem Spieler gegen den Computer (TI) zu spielen.

Bei diesem 17 + 4 werden bei jedem Mitspieler (Partner) Zahlen von 0! bis 11 rein zufällig ausgegeben. Hierzu muß jeder Partner den Joystick(1) nach links oder rechts drücken. Die Alpha-Lock-Taste kann ruhig eingerastet bleiben!
Der Speech-Synthesizer kommt wie auch beim Vorspann oder beim Aufbau der Spielfelder in Aktion. Die einzelnen Zahlen werden optisch sichtbar gemacht und akustisch wiedergegeben. Jede einzelne Zahl wird für jeden Mitspieler sofort addiert. Man hat max. sieben Versuche und muß mind. den Wert 17 erreichen. Hat man diesen Wert noch nicht erreicht und versucht es ein achtes Mal, dann weist der Computer bzw. Speech-Synthesizer darauf hin, daß dies nicht mehr möglich ist. Der nächste Partner soll dann spielen. Wenn man meint, daß man eine akzeptable Zahl (z.B. 19) erreicht hat, drückt man auf die Leertaste, und der nächste Spieler ist an der Reihe. Hat man genau den Wert 21 erreicht, ertönt ein akustisches Signal. Der Synthesizer sagt an, daß man einen Punkt bekommt, und zu den bereits erzielten Punkten wird ein Punkt hinzugezählt. Hat man den Wert von 21 überschritten, dann ertönt ein anderes akustisches Signal, und der Synthesizer sagt an, daß der entsprechende Spieler nicht gewonnen hat. Bewegt man den Joystick(1) wieder nach links oder rechts, obwohl der Partner den Wert 21 schon erreicht hat, weist der Synthesizer darauf hin, daß ein weiterer Versuch nicht mehr möglich ist und der nächste Partner spielen soll. Wird weder der Joystick(1) noch die Leertaste benutzt, wird der entsprechende Mitspieler optisch und akustisch darauf aufmerksam gemacht, daß dieser tätig werden soll. Hat man den Wert 17 noch nicht erreicht und drückt auf die Leertaste, dann weist der Synthesizer darauf hin, daß der nächste Partner noch nicht spielen kann, da der eben erwähnte Wert noch nicht erreicht wurde. Wenn alle Partner gespielt haben, kommt der TI zum Zuge.
Er vergleicht, während seiner Würfe, den besten Wert eines Mitspielers und versucht, diesen zu erreichen. Hierbei muß der TI ebenfalls mind. den Wert 17 erzielen. Schafft er dies bei sieben Versuchen nicht, so beginnt er wieder von vorne. Hat er den Wert 17 im siebten Versuch erreicht und einer der Partner hat z.B. den Wert 19, so beginnt er ebenfalls wieder von vorne. Hat er den gleichen oder den besseren Wert, der kleiner als 22 sein muß, als die Mitspieler erzielt, bekommt er durch die Anzahl der Spieler entsprechend Punkte. Hat er verloren, bekommen die entsprechenden Partner, die mind. den Wert 17 und max. den Wert 21 haben, je einen Punkt.
Nach jedem Durchgang wird überprüft, ob die am Anfang vorgegebene Punktezahl (mind. 10, max. 99) der TI oder die Mannschaft (alle Partner) erreicht haben. Ist dies nicht der Fall, beginnt ein neuer Durchgang oder das Spiel ist damit beendet.
Es erfolgt eine Auflistung der vom TI und der einzelnen Partner erzielten Punkte. Übersichtshalber werden die Punkte der einzelnen Partner addiert und nochmals extra angezeigt.
Zum Schluß wird gefragt, ob man noch einmal spielen möchte oder nicht.

LISTINGS

```
10Ø ! ***********************
11 ! *                     *
12 ! *  BITTE KOPF NICHT   *
13 ! *    MIT EINGEBEN     *
14 ! *                     *
15 ! ***********************
16 ! *                     *
17 ! *        17 + 4       *
18 ! *                     *
19 ! *    Copyright by     *
2Ø ! *                     *
21 ! *   Manfred Polzin    *
22 ! *                     *
23 ! *                     *
24 ! * Benoetigte Geraete  *
25 ! *                     *
26 ! *  TI99/4A-Konsole    *
27 ! *    +Disk-station    *
28 ! *      +Ext.Basic     *
29 ! *+Speech-Synthesizer  *
3Ø ! *     +Joystick(1)    *
31 ! *                     *
32 ! *  Speicherbelegung   *
33 ! *     1Ø42Ø Bytes     *
34 ! *                     *
35 ! ***********************
1ØØ CALL CLEAR :: CALL SCREE
N(16):: ON WARNING NEXT :: C
ALL MAGNIFY(2):: OPTION BASE
 1 :: DIM SP$(8),PUNKTE(8),P
UB(8)
11Ø CALL SPRITE(#1,49,5,13,8
5,#2,55,5,13,1Ø1,#3,43,5,13,
125,#4,52,5,13,149)
12Ø CALL SAY("ONE OF THE GOO
D GAMES IS SEVEN+TEEN AND FO
UR"):: FOR I=1 TO 2ØØ :: NEX
T I
13Ø DISPLAY AT(6,9):"(C) 1Ø/
1986" :: DISPLAY AT(8,6):"VO
N MANFRED POLZIN" :: DISPLAY
 AT(11,1):"ANZAHL UND NAMEN
DER SPIELER"
14Ø DISPLAY AT(12,1):RPT$("?
",28):: DISPLAY AT(24,2):"TI
-99ER WORKSHOP RHEINLAND" ::
 DISPLAY AT(13,1):"ANZAHL?"
:: ACCEPT AT(13,9)VALIDATE("
12345678")SIZE(1)BEEP:SP
15Ø DISPLAY AT(13,1Ø):",GES.
PUNKTEZAHL?" :: ACCEPT AT(13
,27)VALIDATE(DIGIT)BEEP:GES
:: IF GES<1Ø THEN 15Ø
16Ø FOR I=1 TO SP :: DISPLAY
 AT(14+I,7):"NAME";I;"?" ::
ACCEPT AT(14+I,16)BEEP:SP$(I
):: NEXT I
17Ø CALL CLEAR :: FOR A=1 TO
 1Ø :: READ B,C :: CALL COLO
R(B,C,1):: NEXT A
18Ø DATA 3,2,4,2,5,5,6,5,7,7
,8,7,9,13,1Ø,13,11,11,12,11
19Ø CALL CHAR(33,RPT$("ØC",8
),34,"FFFFFF",35,"ØØØØØØØØØØ
FFFFFF",36,RPT$("3Ø",8))
2ØØ CALL CHAR(37,"FCFCFCØCØC
ØCØCØC",38,"3F3F3F3Ø3Ø3Ø3Ø3Ø
",39,"ØCØCØCØCØCFCFCFC",4Ø,"
3Ø3Ø3Ø3Ø3Ø3F3F3F")
21Ø CALL CHAR(41,"ØCØCØCFCFC
ØCØCØC",42,"3Ø3Ø3Ø3F3F3Ø3Ø3Ø
",44,"ØØØØØØFFFF",46,"FFFFFF
ØCØCØCØCØC",47,"ØCØCØCØCØCFF
FFFF")
22Ø CALL CHAR(43,"ØØ18187E7E
1818",45,"ØCØCØCFFFFØCØCØC",
61,"ØØØØ7C7CØØ7C7CØØ",63,"3C
7E66Ø6ØC18ØØ18")
23Ø CALL CHAR(139,"3C3C18181
8183C3C",138,"7E7E1818181818
18",143,"18183C7E5A183C66")
24Ø FOR A=1 TO 12 :: READ D,
E,F,G,H,A$ :: CALL CHAR(D,A$
,E,A$,F,A$,G,A$,H,A$):: NEXT
 A
25Ø DATA 48,64,8Ø,96,112,7CF
ECEDEF6E6FE7C
26Ø DATA 49,65,81,97,113,183
8787818181818
27Ø DATA 5Ø,66,82,98,114,7CF
EØE3C78EØFEFE
28Ø DATA 51,67,83,99,115,7CF
EØE3C3CØEFE7C
29Ø DATA 52,68,84,1ØØ,116,1C
3C7CECFEFEØCØC
3ØØ DATA 53,69,85,1Ø1,117,FE
FECØFCFEØ6FE7C
31Ø DATA 54,7Ø,86,1Ø2,118,7C
FECØFCFEC6FE7C
32Ø DATA 55,71,87,1Ø3,119,FE
FEØ6ØE1C383Ø3Ø
33Ø DATA 56,72,88,1Ø4,12Ø,7C
FEC67C7CC6FE7C
34Ø DATA 57,73,89,1Ø5,121,7C
FEC6FE7EØ6FE7C
35Ø DATA 6Ø,76,92,1Ø8,124,DE
DED2D2D2D2DEDE
36Ø DATA 62,78,94,11Ø,126,66
66666666666666
37Ø DISPLAY AT(3,25):GES ::
GOSUB 55Ø :: ZE=5 :: FOR I=1
 TO SP+1 :: ON I GOSUB 4ØØ,4
1Ø,42Ø,43Ø,44Ø,45Ø,46Ø,47Ø,4
8Ø
38Ø GOSUB 49Ø :: NEXT I :: G
OSUB 57Ø
39Ø FOR I=1 TO 8ØØ :: NEXT I
 :: GOTO 59Ø
4ØØ CALL SPRITE(#5,138,2,35,
33,#6,139,2,35,49):: CALL SA
Y("#TEXAS INSTRUMENTS"):: CA
```

```
LL SAY("HOME+COMPUTER"):: RE
TURN
410 CALL SPRITE(#7,143,5,53,
33,#8,49,5,53,49):: CALL SAY
("PARTNER ONE"):: RETURN
420 CALL SPRITE(#9,143,7,69,
33,#10,50,7,69,49):: CALL SA
Y("PARTNER TWO"):: RETURN
430 CALL SPRITE(#11,143,13,8
5,33,#12,51,13,85,49):: CALL
 SAY("PARTNER THREE"):: RETU
RN
440 CALL SPRITE(#13,143,11,1
01,33,#14,52,11,101,49):: CA
LL SAY("PARTNER FOUR"):: RET
URN
450 CALL SPRITE(#15,143,5,11
7,33,#16,53,5,117,49):: CALL
 SAY("PARTNER FIVE"):: RETUR
N
460 CALL SPRITE(#17,143,7,13
3,33,#18,54,7,133,49):: CALL
 SAY("PARTNER SIX"):: RETURN
470 CALL SPRITE(#19,143,13,1
49,33,#20,55,13,149,49):: CA
LL SAY("PARTNER SEVEN"):: RE
TURN
480 CALL SPRITE(#21,143,11,1
65,33,#22,56,11,165,49):: CA
LL SAY("PARTNER EIGHT"):: RE
TURN
490 ZE=ZE+1 :: CALL HCHAR(ZE
,9,36)
500 FOR J=11 TO 21 STEP 2 ::
 CALL HCHAR(ZE,J,43):: NEXT
J
510 CALL HCHAR(ZE,23,33):: C
ALL HCHAR(ZE,24,61):: DISPLA
Y AT(ZE,23)SIZE(2):"00" :: C
ALL HCHAR(ZE,27,33):: DISPLA
Y AT(ZE,26)SIZE(2):"00"
520 CALL HCHAR(ZE,30,33):: C
ALL HCHAR(ZE+1,9,42):: CALL
HCHAR(ZE+1,10,44,13):: CALL
HCHAR(ZE+1,23,45)
530 CALL HCHAR(ZE+1,24,44,3)
:: CALL HCHAR(ZE+1,27,45)::
CALL HCHAR(ZE+1,28,44,2):: C
ALL HCHAR(ZE+1,30,41):: ZE=Z
E+1
540 RETURN
550 CALL HCHAR(5,9,38):: CAL
L HCHAR(5,10,34,13):: CALL H
CHAR(5,23,46):: CALL HCHAR(5
,24,34,3):: CALL HCHAR(5,27,
46):: CALL HCHAR(5,28,34,2)
560 CALL HCHAR(5,30,37):: RE
TURN
570 CALL HCHAR(ZE,9,40):: CA
LL HCHAR(ZE,10,35,13):: CALL
 HCHAR(ZE,23,47):: CALL HCHA
```

LISTINGS

```
R(ZE,24,35,3):: CALL HCHAR(Z
E,27,47)
580 CALL HCHAR(ZE,28,35,2)::
 CALL HCHAR(ZE,30,39):: RETU
RN
590 FOR Q=1 TO SP :: PUNKTE(
Q)=0 :: NEXT Q :: Z=0
600 FOR I=Z+1 TO SP :: SPA=8
610 FOR J=1 TO 250
620 CALL JOYST(1,X,Y):: CALL
 KEY(0,K,S):: IF K=32 AND PU
NKTE(I)<17 AND SPA<22 THEN G
OSUB 770 :: GOTO 610
630 IF (X=4 AND Y=0)OR(X=-4
AND Y=0)THEN GOSUB 670 ELSE
IF K=32 THEN GOSUB 750 :: GO
SUB 720 :: I=I+1 :: SPA=8 ::
 GOSUB 670
640 NEXT J
650 ON I GOSUB 860,870,880,8
90,900,910,920,930
660 GOTO 610
670 IF PUNKTE(I)>20 THEN 830
680 GOSUB 1110
690 ON I GOSUB 1070,1080,109
0,1100,1070,1080,1090,1100
700 GOSUB 980
710 GOTO 610
720 IF I=0 THEN 1850
730 RETURN
740 NEXT I
750 IF I=SP THEN I=0 :: Z=-1
760 RETURN
770 GOSUB 1110
780 CALL SAY("#"):: CALL SAY
("THE+NEXT+PARTNER+CAN+NOT+P
LAY"):: CALL SAY("YOU+HAVE+N
O+COME+UP+TO+THE+SEVEN+TEEN"
):: CALL SAY("MOVE+THE+JOYST
ICK+ONE"):: RETURN
790 CALL SAY("I+CLEAR+NUMBER
+FOR+NUMBER"):: FOR A=1 TO S
P :: DISPLAY AT(A*2+6,23)SIZ
E(2):"00" :: PUNKTE(A)=0
800 FOR I1=10 TO 22 STEP 2 :
: CALL HCHAR(A*2+6,I1,32)::
NEXT I1 :: NEXT A
810 FOR Q=10 TO 22 STEP 2 ::
 CALL HCHAR(6,Q,32):: NEXT Q
 :: DISPLAY AT(6,23)SIZE(2):
"00" :: CALL HCHAR(3,25,32,2
)
820 FOR P=1 TO 300 :: NEXT P
 :: Z=-1 :: GOTO 590
830 GOSUB 1110
840 CALL SAY("#"):: CALL SAY
("YOU+HAVE+TWENTY+ONE+OR+MOR
E"):: CALL SAY("PRESS+THE+SP
ACE+KEY+FOR+THE+NEXT+PARTNER
"):: GOTO 610
```

LISTINGS

```
850 CALL SAY("YOU+MUST+MOVE+
THE+JOYSTICK+ONE+FOR+THE+NEX
T+NUMBER"):: CALL SAY("OR+PR
ESS+THE+SPACE+KEY+FOR+THE+NE
XT+PARTNER"):: RETURN
860 GOSUB 940 :: GOSUB 410 :
: GOSUB 850 :: RETURN
870 GOSUB 940 :: GOSUB 420 :
: GOSUB 850 :: RETURN
880 GOSUB 940 :: GOSUB 430 :
: GOSUB 850 :: RETURN
890 GOSUB 940 :: GOSUB 440 :
: GOSUB 850 :: RETURN
900 GOSUB 940 :: GOSUB 450 :
: GOSUB 850 :: RETURN
910 GOSUB 940 :: GOSUB 460 :
: GOSUB 850 :: RETURN
920 GOSUB 940 :: GOSUB 470 :
: GOSUB 850 :: RETURN
930 GOSUB 940 :: GOSUB 480 :
: GOSUB 850 :: RETURN
940 FOR W=1 TO 5 :: CALL SPR
ITE(#I*2+5,63,2,I*16+37,33,#
I*2+6,63,2,I*16+37,49):: CAL
L SOUND(-250,330,6,660,6,990
,6)
950 FOR W1=1 TO 100 :: NEXT
W1
960 CALL SPRITE(#I*2+5,143,2
,I*16+37,33,#I*2+6,I+48,2,I*
16+37,49)
970 CALL SOUND(-250,110,6,22
0,6,330,6):: FOR W1=1 TO 100
 :: NEXT W1 :: NEXT W :: FOR
 W1=1 TO 200 :: NEXT W1 :: R
ETURN
980 SPA=SPA+2 :: CALL HCHAR(
I*2+6,SPA,COD):: PUNKTE(I)=P
UNKTE(I)+PU :: DISPLAY AT(I*
2+6,23)SIZE(2):USING "##":PU
NKTE(I)
990 IF PUNKTE(I)>21 AND SPA<
23 THEN GOSUB 1050 ELSE IF P
UNKTE(I)=21 AND SPA<23 THEN
GOSUB 1020
1000 IF SPA>22 THEN CALL SAY
("#"):: CALL SAY("THE+NEXT+P
ARTNER+MUST+PLAY"):: GOSUB 1
110 :: CALL SAY("PRESS+THE+S
PACE+KEY"):: GOSUB 1060
1010 RETURN
1020 FOR SO=330 TO 660 STEP
5 :: CALL SOUND(-100,SO,6,SO
+110,6,SO+220,6):: NEXT SO :
: GOSUB 1110 :: CALL SAY("YO
U+GET+ONE+POINT+TO+THAT")
1030 PU8(I)=PU8(I)+1 :: DISP
LAY AT(I*2+6,26)SIZE(2)BEEP:
PU8(I):: IF PU8(I)<10 THEN D
ISPLAY AT(I*2+6,26)SIZE(1):"
0"
1040 RETURN
1050 FOR SO=550 TO 220 STEP
-5 :: CALL SOUND(-100,SO,6,S
O+110,6,SO+220,6):: NEXT SO
:: GOSUB 1110 :: CALL SAY("Y
OU+HAVE+NOT+WON"):: RETURN
1060 PUNKTE(I)=PUNKTE(I)-PU
:: DISPLAY AT(I*2+6,23)SIZE(
2):USING "##":PUNKTE(I):: CA
LL HCHAR(I*2+6,SPA,61):: SPA
=SPA-2 :: RETURN
1070 RANDOMIZE :: ZUFALL1=IN
T(RND*12)+1 :: ON ZUFALL1 GO
TO 1130,1140,1150,1160,1170,
1180,1190,1200,1210,1220,123
0,1240
1080 RANDOMIZE :: ZUFALL2=IN
T(RND*12)+1 :: ON ZUFALL2 GO
TO 1250,1260,1270,1280,1290,
1300,1310,1320,1330,1340,135
0,1360
1090 RANDOMIZE :: ZUFALL3=IN
T(RND*12)+1 :: ON ZUFALL3 GO
TO 1370,1380,1390,1400,1410,
1420,1430,1440,1450,1460,147
0,1480
1100 RANDOMIZE :: ZUFALL4=IN
T(RND*12)+1 :: ON ZUFALL4 GO
TO 1490,1500,1510,1520,1530,
1540,1550,1560,1570,1580,159
0,1600
1110 ON I GOSUB 410,420,430,
440,450,460,470,480
1120 RETURN
1130 COD=64 :: GOSUB 1730 ::
 RETURN
1140 COD=65 :: GOSUB 1740 ::
 RETURN
1150 COD=66 :: GOSUB 1750 ::
 RETURN
1160 COD=67 :: GOSUB 1760 ::
 RETURN
1170 COD=68 :: GOSUB 1770 ::
 RETURN
1180 COD=69 :: GOSUB 1780 ::
 RETURN
1190 COD=70 :: GOSUB 1790 ::
 RETURN
1200 COD=71 :: GOSUB 1800 ::
 RETURN
1210 COD=72 :: GOSUB 1810 ::
 RETURN
1220 COD=73 :: GOSUB 1820 ::
 RETURN
1230 COD=76 :: GOSUB 1830 ::
 RETURN
1240 COD=78 :: GOSUB 1840 ::
 RETURN
1250 COD=80 :: GOSUB 1730 ::
 RETURN
```

LISTINGS

```
1260 COD=81 :: GOSUB 1740 ::
 RETURN
1270 COD=82 :: GOSUB 1750 ::
 RETURN
1280 COD=83 :: GOSUB 1760 ::
 RETURN
1290 COD=84 :: GOSUB 1770 ::
 RETURN
1300 COD=85 :: GOSUB 1780 ::
 RETURN
1310 COD=86 :: GOSUB 1790 ::
 RETURN
1320 COD=87 :: GOSUB 1800 ::
 RETURN
1330 COD=88 :: GOSUB 1810 ::
 RETURN
1340 COD=89 :: GOSUB 1820 ::
 RETURN
1350 COD=92 :: GOSUB 1830 ::
 RETURN
1360 COD=94 :: GOSUB 1840 ::
 RETURN
1370 COD=96 :: GOSUB 1730 ::
 RETURN
1380 COD=97 :: GOSUB 1740 ::
 RETURN
1390 COD=98 :: GOSUB 1750 ::
 RETURN
1400 COD=99 :: GOSUB 1760 ::
 RETURN
1410 COD=100 :: GOSUB 1770 :
: RETURN
1420 COD=101 :: GOSUB 1780 :
: RETURN
1430 COD=102 :: GOSUB 1790 :
: RETURN
1440 COD=103 :: GOSUB 1800 :
: RETURN
1450 COD=104 :: GOSUB 1810 :
: RETURN
1460 COD=105 :: GOSUB 1820 :
: RETURN
1470 COD=108 :: GOSUB 1830 :
: RETURN
1480 COD=110 :: GOSUB 1840 :
: RETURN
1490 COD=112 :: GOSUB 1730 :
: RETURN
1500 COD=113 :: GOSUB 1740 :
: RETURN
1510 COD=114 :: GOSUB 1750 :
: RETURN
1520 COD=115 :: GOSUB 1760 :
: RETURN
1530 COD=116 :: GOSUB 1770 :
: RETURN
1540 COD=117 :: GOSUB 1780 :
: RETURN
1550 COD=118 :: GOSUB 1790 :
: RETURN
1560 COD=119 :: GOSUB 1800 :
: RETURN
1570 COD=120 :: GOSUB 1810 :
: RETURN
1580 COD=121 :: GOSUB 1820 :
: RETURN
1590 COD=124 :: GOSUB 1830 :
: RETURN
1600 COD=126 :: GOSUB 1840 :
: RETURN
1610 COD=48 :: GOSUB 1730 ::
 RETURN
1620 COD=49 :: GOSUB 1740 ::
 RETURN
1630 COD=50 :: GOSUB 1750 ::
 RETURN
1640 COD=51 :: GOSUB 1760 ::
 RETURN
1650 COD=52 :: GOSUB 1770 ::
 RETURN
1660 COD=53 :: GOSUB 1780 ::
 RETURN
1670 COD=54 :: GOSUB 1790 ::
 RETURN
1680 COD=55 :: GOSUB 1800 ::
 RETURN
1690 COD=56 :: GOSUB 1810 ::
 RETURN
1700 COD=57 :: GOSUB 1820 ::
 RETURN
1710 COD=60 :: GOSUB 1830 ::
 RETURN
1720 COD=62 :: GOSUB 1840 ::
 RETURN
1730 PU=0 :: CALL SAY("ZERO"
):: RETURN
1740 PU=1 :: CALL SAY("ONE")
:: RETURN
1750 PU=2 :: CALL SAY("TWO")
:: RETURN
1760 PU=3 :: CALL SAY("THREE
"):: RETURN
1770 PU=4 :: CALL SAY("FOUR"
):: RETURN
1780 PU=5 :: CALL SAY("FIVE"
):: RETURN
1790 PU=6 :: CALL SAY("SIX")
:: RETURN
1800 PU=7 :: CALL SAY("SEVEN
"):: RETURN
1810 PU=8 :: CALL SAY("EIGHT
"):: RETURN
1820 PU=9 :: CALL SAY("NINE"
):: RETURN
1830 PU=10 :: CALL SAY("TEN"
):: RETURN
1840 PU=11 :: CALL SAY("ELEV
EN"):: RETURN
1850 SPAL=8 :: POINTS=0 :: F
OR J1=10 TO 22 STEP 2 :: CAL
L HCHAR(6,J1,32):: NEXT J1 :
```

LISTINGS

```
: DISPLAY AT(6,22)SIZE(3):"=
ØØ"
186Ø FOR L=1 TO 7 :: GOSUB 4
ØØ
187Ø RANDOMIZE :: COMP=INT(R
ND*12)+1 :: ON COMP GOSUB 16
1Ø,162Ø,163Ø,164Ø,165Ø,166Ø,
167Ø,168Ø,169Ø,17ØØ,171Ø,172
Ø
188Ø GOSUB 224Ø
189Ø IF POINTS<17 THEN 193Ø
19ØØ IF POINTS>16 AND POINTS
<21 THEN 195Ø
191Ø IF POINTS=21 THEN 195Ø
192Ø IF POINTS>21 THEN GOSUB
 222Ø :: GOTO 2Ø6Ø
193Ø NEXT L
194Ø GOTO 185Ø
195Ø PUN=Ø :: FOR Z=1 TO SP
:: IF PUNKTE(Z)<22 AND PUNKT
E(Z)>PUN THEN PUNKT=PUN :: P
UN=PUNKTE(Z)
196Ø NEXT Z
197Ø PUNKT=Ø :: FOR Z=1 TO S
P :: IF PUNKTE(Z)<22 AND PUN
KTE(Z)<PUNKT THEN 198Ø ELSE
IF PUNKTE(Z)<22 THEN PUNKT=P
UNKTE(Z)
198Ø NEXT Z
199Ø DISPLAY AT(3,23)SIZE(2)
:USING "##":PUNKT
2ØØØ FOR Z=1 TO 6
2Ø1Ø IF SPAL>22 THEN 185Ø EL
SE IF POINTS=21 THEN 2Ø6Ø EL
SE IF POINTS<21 AND POINTS<P
UNKT AND PUNKT<22 AND SPAL<2
3 THEN GOSUB 4ØØ :: GOTO 2Ø3
Ø
2Ø2Ø GOTO 2Ø5Ø
2Ø3Ø RANDOMIZE :: TI=INT(RND
*12)+1 :: ON TI GOSUB 161Ø,1
62Ø,163Ø,164Ø,165Ø,166Ø,167Ø
,168Ø,169Ø,17ØØ,171Ø,172Ø
2Ø4Ø GOSUB 224Ø
2Ø5Ø NEXT Z
2Ø6Ø @=1 :: FOR Z=1 TO SP
2Ø7Ø IF POINTS=21 AND @=1 TH
EN GOSUB 223Ø :: GOSUB 214Ø
ELSE IF POINTS=21 THEN GOSUB
 214Ø
2Ø8Ø IF POINTS<21 AND POINTS
<PUNKTE(Z)AND PUNKTE(Z)>21 A
ND @=1 THEN GOSUB 223Ø :: GO
SUB 214Ø ELSE IF POINTS<21 A
ND POINTS<PUNKTE(Z)AND PUNKT
E(Z)>21 THEN GOSUB 214Ø
2Ø9Ø IF POINTS<21 AND POINTS
>=PUNKTE(Z)AND PUNKTE(Z)<22
AND @=1 THEN GOSUB 223Ø :: G
OSUB 214Ø ELSE IF POINTS<21
AND POINTS>=PUNKTE(Z)AND PUN
KTE(Z)<22 THEN GOSUB 214Ø
21ØØ IF POINTS>PUNKTE(Z)AND
PUNKTE(Z)<22 AND POINTS>21 T
HEN GOSUB 219Ø
211Ø @=@+1 :: NEXT Z
212Ø GOSUB 225Ø
213Ø FOR P=1 TO 5ØØ :: NEXT
P :: GOTO 218Ø
214Ø CALL SAY("I+GET+ONE+POI
NT+TO+THAT")
215Ø PU9=PU9+1 :: DISPLAY AT
(6,26)SIZE(2)BEEP:USING "##"
:PU9 :: IF PU9<1Ø THEN DISPL
AY AT(6,26)SIZE(1):"Ø"
216Ø RETURN
217Ø FOR P=1 TO 15Ø :: NEXT
P :: GOTO 213Ø
218Ø GOTO 79Ø
219Ø ON Z GOSUB 41Ø,42Ø,43Ø,
44Ø,45Ø,46Ø,47Ø,48Ø
22ØØ CALL SAY("YOU+GET+ONE+P
OINT+TO+THAT"):: PU8(Z)=PU8(
Z)+1 :: DISPLAY AT(Z*2+6,26)
SIZE(2)BEEP:USING "##":PU8(Z
):: IF PU8(Z)<1Ø THEN DISPLA
Y AT(Z*2+6,26)SIZE(1):"Ø"
221Ø RETURN
222Ø CALL SAY("#"):: CALL SA
Y("I+HAVE+NOT+WON"):: RETURN
223Ø CALL SAY("I+STOP"):: CA
LL SAY("I+HAVE+WON"):: RETUR
N
224Ø SPAL=SPAL+2 :: CALL HCH
AR(6,SPAL,COD):: POINTS=POIN
TS+PU :: DISPLAY AT(6,23)SIZ
E(2):USING "##":POINTS :: RE
TURN
225Ø GES8=Ø :: FOR I=1 TO SP
 :: GES8=GES8+PU8(I):: NEXT
I :: IF GES8=PU9 THEN RETURN
 ELSE IF GES8>=GES OR PU9>=G
ES THEN 227Ø
226Ø RETURN
227Ø FOR I=5 TO 22 :: CALL D
ELSPRITE(#I):: NEXT I :: CAL
L CLEAR :: CALL CHARSET :: P
RINT "TEXAS INSTRUMENTS=";PU
9;"PUNKTE"
228Ø PRINT :: PRINT :: FOR I
=1 TO SP :: PRINT SP$(I);"="
;PU8(I);"PUNKTE" :: NEXT I :
: PRINT :: PRINT :: PRINT "G
ESAMTE PUNKTEZAHL =";GES8
229Ø PRINT :: PRINT :: PRINT
23ØØ PRINT "NOCH EINMAL SPIE
LEN (J/N) ?"
231Ø CALL KEY(Ø,K,S):: IF S=
Ø THEN 231Ø ELSE IF K=74 THE
N DISPLAY AT(13,7)ERASE ALL:
"BITTE WARTEN !" :: RUN 1ØØ
ELSE IF K=78 THEN CALL CLEAR
 :: PRINT "* BIS ZUM NAECHST
EN MAL *" :: END ELSE 231Ø
```

```
880 ! SUBPROGRAMME
890 !
900 SUB E(ZE,LA,W)
910 ACCEPT AT(ZE,26-LA)BEEP
SIZE(-LA)VALIDATE(DIGIT):GE
:: ACCEPT AT(ZE,27)SIZE(-2)V
ALIDATE(DIGIT):KE :: W=GE+KE
/100 :: SUBEND
```


Scrabble 86

Bei Scrabble86 handelt es sich um das fast gleichnamige Spiel. Bis zu 4 Spieler können hier mitspielen. Spieler 1 und 3 benutzen Joystick 1 und Spieler 2 und 4 Joystick 2. Die Nameneintragung zu Beginn des Spieles erfolgt im Kleinschreibemodus. Das Leerzeichen wird mit der offenen geschweiften Klammer simuliert. Einen Namen, der nur aus solchen „Leerzeichen" besteht, überspringt das Programm bei der Wortgenerierung. Der Spieler, der momentan dran ist, erhält einen abgesetzten Namenszug und der Cursor zeigt auf seine Buchstaben. Um einen Buchstaben aufzunehmen, muß die Spitze des Cursors über den betreffenden Buchstaben geführt und anschließend der Feuerknopf gedrückt werden. Der Buchstabe wird abgesetzt dargestellt und es kann nun mit dem Joystick zur betreffenden Stelle auf dem Spielbrett gefahren werden. Ein erneuter Druck auf den Feuerknopf plaziert den Buchstaben an der Cursorspitze. Jeder Spieler kann sechs Buchstaben auf dem Spielbrett plazieren. Nachdem ein Buchstabe plaziert wurde, wird an der nun freien Stelle unterhalb des Namens ein neuer Buchstabe dargestellt.

Um einen Buchstaben gegen einen zufällig generierten Buchstaben auszutauschen, muß zuerst 5 gedrückt werden, damit 2 neue Icons erscheinen. Mit dem Cursor wird nun das Icon One Swap angeklickt und der Cursor steht nun in der Reihe der Buchstaben des betreffenden Spielers. Mit dem Joystick kann ein Buchstabe ausgewählt werden und dieser wird dann durch einen neuen ersetzt, wenn der Feuerknopf gedrückt wird. Jeder Spieler kann bei einer Wortgenerierung sechsmal einen Buchstaben tauschen. Um einen schon plazierten Buchstaben wieder zu entfernen, muß das Icon One Back angelickt werden (evtl. vorher 5 drücken). Hierbei wird der Buchstaben, der zuletzt plaziert wurde, wieder entfernt und es wird auch die Punktzahl, die damit erreicht wurde, wieder abgezogen. Die Punktzahl wird um den Wert, der auf dem auserwählten Feld steht, mal 10 addiert. Ein Fragezeichen ergibt eine zufällige Punktzahl zwischen 0 und 100. Wenn kein Buchstabe mehr plaziert werden kann, so muß das Icon No Move angelickt werden, damit der nächste Spieler weitermachen kann. Wenn ein Joker als Buchstaben auserwählt wurde und dieser dann auf dem Spielbrett plaziert wurde, dann kann man auswählen, welcher Buchstabe der Joker werden soll. Es wird einfach die entsprechende Taste auf der Tastatur gedrückt. Um das Spiel zu beenden, muß einfach das Icon End angeklickt werden und das Programm gibt die Plazierung der einzelnen Spieler mit deren Punktzahl aus. Die Leertaste beendet dann das Programm.

```
10 REM********************
11 REM*       PIRAT        *
12 REM*                     *
13 REM*  Copyright   by     *
14 REM*                     *
15 REM*  Michael Keusen     *
16 REM*                     *
17 REM*Benoetigte Geraete*
18 REM* TI99/4A  Konsole *
19 REM*     Joystick 1      *
20 REM*                     *
23 REM* Speicherbelegung *
24 REM*     9848 Bytes      *
25 REM********************
26 REM
100 DATA 159,1818FF999918244
2,97,1010101010101010,98,7EF
FFF7E2A498A46,111,7EFFFF7E2A
498A46
110 DATA 99,1010101010183C7E
,126,3C7EE7FFDBDBFFFF,101,80
80808080808080,102,FF,103,01
01010101010101
120 DATA 104,0000FF18FF81FF1
8,105,FF81FF18FF81FF18,35,00
00000000000000
130 DATA 106,FF81FF81FF81FF8
1,107,FFFFFFFFFFFFFFFF,108,0
204081820408 0FF,109,00FF
140 DATA 113,8080FF18FF81FF1
8,114,0101FF18FF81FF18,115,F
F80808080808080,116,FF010101
01010101
150 DATA 46,0000000000000000
,94,0000E0BEE2,110,402010080
40201FF,32,FFFFFFFFFFFFFFFF,
33,00000,59,0000000000000000
160 DATA 122,BD7E3C3C3C3C3C1
8,123,8040FEFFFFFE4080,124,1
83C3C3C3C3C7EBD,125,01027FFF
FF7F0201
170 DATA 130,1818FF999918244
2,140,0000E0BEE2,131,1818FFF
F18181818,144,4200400BA00085
180 DATA 146,422442003C7EFEF
F,147,244224003C7EFEFF,145,8
500A0B040100042,64,FF81BDA1A
1BD81FF,58,88D8A98A0C0C0A09
190 CALL CLEAR
200 CALL SCREEN(2)
210 RESTORE 100
220 FOR CHAR=1 TO 39
230 READ Z,CHAR$
240 CALL CHAR(Z,CHAR$)
250 NEXT CHAR
260 FOR COLOR=1 TO 11
270 CALL COLOR(COLOR,1,1)
280 NEXT COLOR
290 FOR WA=1 TO 16
300 PRINT "gie!!!!!!!!!!!!!
!!!!!!!gie"
310 NEXT WA
320 PRINT "gie.............
```

```
........gie"
330 PRINT "giqhhhhhhhhhhjhhh
hhhhhhhhriegiiiiiiiiiiejgii
iiiiiiiiiegiiiiiiiiiiejgii
iiiiiiiiie"
340 PRINT "giiiiiiiiiiejgii
iiiiiiiiiegisffffffff!j!ff
fffffffftiegie..^.......j...
........gie"
350 PRINT "giqhhhhhhhhhhjhhn
.lhhhhhhrie"
360 PRINT TAB(17);"!m!"
370 GOSUB 2930
380 FOR COLOR=1 TO 11
390 CALL COLOR(COLOR,2,5)
400 NEXT COLOR
410 CALL COLOR(12,11,5)
420 CALL COLOR(16,10,5)
430 CALL COLOR(3,16,5)
440 CALL COLOR(4,16,5)
450 CALL COLOR(2,16,5)
460 CALL COLOR(15,2,5)
470 MOR=23
480 VC=0
490 NRW=24
500 ITT=0
510 LU=5
520 AT=7
530 AC=125
540 ABO=21
550 SCHRITT=1000
560 RA=1
570 R=1
580 SPI=13
590 SI=21
600 SO=16
610 PUNKTE=0
620 SE=28
630 RWE=1
640 MI=0
650 ZEI$="PUNKTE>"
660 U=3
670 Z=6
680 GOSUB 2670
690 GOSUB 3390
700 CALL COLOR(13,10,1)
710 CALL COLOR(14,15,1)
720 HT=15
730 VT=6
740 CALL JOYST(1,X,Y)
750 CALL HCHAR(HT,VT,159)
760 IF X>0 THEN 850
770 IF X<0 THEN 900
780 IF Y<0 THEN 950
790 IF Y>0 THEN 1000
800 IF RA=1 THEN 840
810 GOSUB 2250
820 SCHRITT=SCHRITT-1
830 GOSUB 2440
840 GOTO 740
850 VT=VT+1
860 R=1
870 GOSUB 1050
880 CALL HCHAR(HT,VT-1,46)
890 GOTO 740
900 VT=VT-1
910 R=2
920 GOSUB 1050
930 CALL HCHAR(HT,VT+1,46)
940 GOTO 740
950 HT=HT+1
960 R=3
970 GOSUB 1050
980 CALL HCHAR(HT-1,VT,46)
990 GOTO 740
1000 HT=HT-1
1010 R=4
1020 GOSUB 1050
1030 CALL HCHAR(HT+1,VT,46)
1040 GOTO 740
1050 IF RA=1 THEN 1200
1060 IF CD<NRW THEN 1130
1070 ON RWE GOTO 1080,1100
1080 RESTORE 2630
1090 GOTO 1110
1100 RESTORE 2890
1110 CD=1
1120 CALL GCHAR(AT,ABO,VC)
1130 CALL HCHAR(AT,ABO,46)
1140 IF VC=159 THEN 3010
1150 READ AT,ABO,AC
1160 CALL GCHAR(AT,ABO,VC)
1170 CALL HCHAR(AT,ABO,AC)
1180 IF VC=159 THEN 3010
1190 CD=CD+1
1200 CALL GCHAR(HT,VT,WB)
1210 GOSUB 1260
1220 IF RA=1 THEN 1250
1230 GOSUB 2250
1240 GOSUB 2440
1250 RETURN
1260 IF WB=94 THEN 1350
1270 IF WB=35 THEN 2710
1280 IF WB=109 THEN 1810
1290 IF WB=106 THEN 1510
1300 IF WB=126 THEN 3600
1310 IF WB=98 THEN 3210
1320 IF WB=111 THEN 3470
1330 IF WB<>46 THEN 1720
1340 RETURN
1350 IF RA=2 THEN 1380
1360 CALL HCHAR(HT,VT+1,46)
1370 GOTO 1390
1380 CALL HCHAR(HT,VT-1,46)
1390 CALL HCHAR(HT,VT,159)
1400 FOR SA=1 TO 5
1410 FOR OO=110 TO 990 STEP
110
1420 CALL SOUND(75,OO,0)
1430 NEXT OO
1440 NEXT SA
1450 PUNKTE=PUNKTE+1000
1460 MI=2
1470 UU=UU+1
1480 GOSUB 3390
1490 GOSUB 2970
```

LISTINGS

```
1500 RETURN
1510 CALL HCHAR(HT-1,VT,46)
1520 FOR MA=1 TO 6
1530 CALL HCHAR(HT,VT,159)
1540 FOR T=1 TO 100
1550 NEXT T
1560 CALL HCHAR(HT,VT,106)
1570 IF MA=4 THEN 1690
1580 HT=HT+1
1590 NEXT MA
1600 HT=HT-1
1610 VT=VT+1
1620 CALL HCHAR(HT,VT,159)
1630 FOR T=1 TO 150
1640 NEXT T
1650 CALL VCHAR(17,16,33,4)
1660 CALL HCHAR(22,16,104)
1670 CALL HCHAR(HT,VT-1,46)
1680 GOTO 740
1690 CALL HCHAR(16,16,104)
1700 CALL SOUND(50,3000,0,30
00,0,3000,0)
1710 GOTO 1580
1720 ON R GOTO 1730,1750,177
0,1790
1730 VT=VT-1
1740 GOTO 740
1750 VT=VT+1
1760 GOTO 740
1770 HT=HT-1
1780 GOTO 740
1790 HT=HT+1
1800 GOTO 740
1810 CALL HCHAR(HT-1,VT,33)
1820 CALL HCHAR(U,Z,33,15)
1830 FOR MA=1 TO 8
1840 PRINT TAB(17);"!!!''''
'"

1850 CALL HCHAR(22,20,159)
1860 CALL HCHAR(23,20,109)
1870 FOR T=1 TO 50
1880 NEXT T
1890 CALL HCHAR(22,20,33)
1900 CALL HCHAR(23,20,33)
1910 NEXT MA
1920 CALL HCHAR(22,20,159)
1930 CALL HCHAR(23,20,109)
1940 PRINT "................
............"
1950 PRINT "'''''e.g''''''''
'''''''''''''''''e.g'''e...
.}.g''''''''''''''e.g'''e.g'
'e.....^.''''''''e........g"
1960 PRINT "''''''''e.g''''
''a''''''''''e.........g''''
!!c!!'''''''e.g'''''''''''''
!!b!!''''''''"
1970 PRINT "e.g''''''''''''''
!!!!!''''a"
1980 PRINT "e...............
.....g''!c!'''''''''''''''''
''''.g''!o!''''''''''''''''''
'e.....g!!!"
1990 PRINT "''''''''''''''''''
'e.'''....^"
2000 PRINT "''''''''''''''''''
'e...g"
2010 PRINT "''''e...........
....sffft'''''''e.g'''''''''
'''.e.~.g''''''''e......^g'''
'''.....g": : :
2020 RANDOMIZE
2030 HILDE=INT(RND*24)+1
2040 IF HILDE<20 THEN 2030
2050 HAANI=INT(RND*32)+1
2060 CALL GCHAR(HILDE,HAANI,
XXX)
2070 IF XXX<>46 THEN 2030
2080 CALL HCHAR(HILDE,HAANI,
35)
2090 HT=3
2100 CALL HCHAR(4,20,46)
2110 RA=2
2120 GOSUB 2930
2130 FOR COLOR=3 TO 7
2140 CALL COLOR(COLOR,16,2)
2150 NEXT COLOR
2160 CALL HCHAR(1,30,64)
2170 CALL HCHAR(1,32,58)
2180 ZEI$="PUNKTE>"
2190 GOSUB 2670
2200 GOSUB 3390
2210 GOSUB 2970
2220 WC=3
2230 RETURN
2240 GOTO 740
2250 IF SU=2 THEN 2360
2260 SPI=SPI+1
2270 CALL VCHAR(SPI-3,SI,97,
2)
2280 CALL HCHAR(SPI-1,SI,99)
2290 CALL GCHAR(SPI,SI,VB)
2300 CALL HCHAR(SPI,SI,98)
2310 IF VB=159 THEN 3210
2320 IF SPI=14 THEN 2340
2330 RETURN
2340 SU=2
2350 RETURN
2360 SPI=SPI-1
2370 CALL HCHAR(SPI+1,SI,46)
2380 CALL HCHAR(SPI-1,SI,99)
2390 CALL HCHAR(SPI,SI,98)
2400 IF SPI=12 THEN 2420
2410 RETURN
2420 SU=1
2430 RETURN
2440 IF SW=2 THEN 2550
2450 SO=SO+1
2460 CALL VCHAR(SO-3,SE,97,2
)
2470 CALL HCHAR(SO-1,SE,99)
2480 CALL GCHAR(SO,SE,VA)
2490 CALL HCHAR(SO,SE,111)
2500 IF VA=159 THEN 3470
2510 IF SO=17 THEN 2530
2520 RETURN
```

```
2530 SW=2
2540 RETURN
2550 SO=SO-1
2560 CALL HCHAR(SO+1,SE,46)
2570 CALL HCHAR(SO-1,SE,99)
2580 CALL HCHAR(SO,SE,111)
2590 IF SO=15 THEN 2610
2600 RETURN
2610 SW=1
2620 RETURN
2630 DATA 7,20,125,7,19,125,
7,18,125,7,17,125,7,16,122,8
,16,122,9,16,125,9,15,125
2640 DATA 9,14,125,9,13,125,
9,12,125,9,11,125,9,10,125,9
,11,123,9,12,123
2650 DATA 9,13,123,9,14,123,
9,15,123,9,16,124,8,16,124,7
,16,124,7,17,123,7,18,123
2660 DATA 7,19,123,7,20,123
2670 FOR AB=1 TO LEN(ZEI$)
2680 CALL HCHAR(U,(Z+AB)-1,A
SC(SEG$(ZEI$,AB,1)))
2690 NEXT AB
2700 RETURN
2710 FOR KAPUTT=1 TO 30
2720 CALL HCHAR(AT,ABO,144)
2730 CALL SOUND(175,-6,KAPUT
T)
2740 CALL HCHAR(AT,ABO,145)
2750 NEXT KAPUTT
2760 CALL COLOR(15,2,5)
2770 FOR KAPUTT=1 TO 40
2780 CALL HCHAR(AT,ABO,146)
2790 CALL HCHAR(AT,ABO,147)
2800 NEXT KAPUTT
2810 PUNKTE=PUNKTE*2
2820 GOSUB 3390
2830 CALL HCHAR(AT,ABO,46)
2840 CD=90
2850 RWE=2
2860 NRW=34
2870 CALL HCHAR(HILDE,HAANI,
46)
2880 GOTO 1720
2890 DATA 18,23,125,18,22,12
5,18,21,122,19,21,122,19,20,
125,19,19,125,19,18,125,19,1
7,125
2900 DATA 19,16,125,19,15,12
5,19,14,125,19,13,125,19,12,
125,19,11,125,19,10,125,19,9
,125,19,8,125
2910 DATA 19,8,123,19,9,123,
19,10,123,19,11,123,19,12,12
3,19,13,123,19,14,123,19,15,
123
2920 DATA 19,16,123,19,17,12
3,19,18,123,19,19,123,19,20,
123,19,21,124,18,21,124,18,2
2,123,18,23,123
2930 CALL HCHAR(24,5,130)
2940 CALL HCHAR(24,7,130)
2950 CALL HCHAR(24,9,130)
2960 RETURN
2970 IF MI=0 THEN 3000
2980 CALL HCHAR(24,MOR,140)
2990 MOR=MOR+2
3000 RETURN
3010 FOR SAU=1 TO 3
3020 FOR SEU=1500 TO 2500 ST
EP 250
3030 CALL SOUND(-1,SEU,0,-1,
0)
3040 NEXT SEU
3050 NEXT SAU
3060 CALL SOUND(-1,-6,0)
3070 CALL CHAR(130,"1818FFFF
18181818")
3080 PUNKTE=PUNKTE/10*2
3090 GOSUB 3390
3100 CALL CHAR(159,"1818FFFF
18181818")
3110 CALL HCHAR(AT,ABO,159)
3120 FOR T=1 TO 20
3130 CALL SOUND(-1,-2,10,300
0-T,10,110*T,5,234,1)
3140 NEXT T
3150 FOR T=2 TO 20
3160 CALL SOUND(10,-2,0)
3170 CALL COLOR(16,T/2,5)
3180 NEXT T
3190 GOTO 3780
3200 REM
3210 FOR T=1 TO 2
3220 FOR SSW=3000 TO 2000 ST
EP -50
3230 CALL SOUND(-1,SSW,0,-8,
0)
3240 NEXT SSW
3250 NEXT T
3260 GOSUB 3340
3270 FOR T=0 TO 30
3280 CALL SOUND(-1,-3,T)
3290 NEXT T
3300 CALL HCHAR(HT,VT-2,46,4
)
3310 VT=20
3320 HT=3
3330 GOTO 2340
3340 CALL HCHAR(24,LU,131)
3350 LU=LU+2
3360 PUNKTE=PUNKTE-330
3370 GOSUB 3390
3380 RETURN
3390 KUN$=";;;;;;"
3400 GOSUB 3420
3410 KUN$=STR$(PUNKTE)
3420 FOR POINT=1 TO LEN(KUN$
)
3430 CALL HCHAR(3,(13+POINT)
-1,ASC(SEG$(KUN$,POINT,1)))
3440 NEXT POINT
3450 CALL SOUND(1,3000,0)
3460 RETURN
3470 FOR T=1 TO 2
```

»

```
3480 FOR SSW=3000 TO 2000 STEP -50
3490 CALL SOUND(-1000,SSW,0,-8,0,SSW+1000,0)
3500 NEXT SSW
3510 NEXT T
3520 GOSUB 3340
3530 FOR T=0 TO 30
3540 CALL SOUND(-100,-3,T)
3550 NEXT T
3560 CALL HCHAR(HT,VT,46)
3570 VT=20
3580 HT=3
3590 GOTO 2530
3600 IF UU=4 THEN 3620
3610 GOTO 1310
3620 CALL CLEAR
3630 ZEI$="BRAVO DU HAST ES GESCHAFFT"
3640 U=12
3650 Z=4
3660 GOSUB 2670
3670 FOR YOU=1 TO 5
3680 FOR T=220 TO 1100 STEP 110
3690 CALL SOUND(-1,T,10,T/2,YOU*6)
3700 NEXT T
3710 NEXT YOU
3720 PRINT : : :;PUNKTE*2+SCHRITT;"PUNKTE"
3730 GESF=GESF+1
3740 PUNKTE=PUNKTE*2+SCHRITT
3750 FOR T=1 TO 500
3760 NEXT T
3770 GOTO 3840
3780 ZEI$="G A M E'''O VaE R"
3790 U=10
3800 Z=8
3810 GOSUB 2670
3820 FOR T=1 TO 500
3830 NEXT T
3840 ZEI$="NOCHMAL <J ODER N>"
3850 U=16
3860 Z=2
3870 GOSUB 2670
3880 CALL KEY(0,G,S)
3890 IF S=0 THEN 3880
3900 IF G=74 THEN 3970
3910 IF G=110 THEN 3980
3920 IF G=106 THEN 3970
3930 IF G=78 THEN 3980
3940 CALL SOUND(-1,1000,0)
3950 GOTO 3880
3960 REM  HIGHSCORE
3970 ITT=9
3980 IF PUNKTE<HPUNKTE THEN 4150
3990 CALL CLEAR
4000 ZEI$="NEUER''HIGHSCORE"
4010 U=2
4020 Z=8
4030 GOSUB 2670
4040 CALL HCHAR(U+1,Z,61,16)
4050 ZEI$="PUNKTE"
4060 U=11
4070 Z=4
4080 GOSUB 2670
4090 ZEI$=STR$(PUNKTE)
4100 U=13
4110 GOSUB 2670
4120 FOR T=1 TO 500
4130 NEXT T
4140 HPUNKTE=PUNKTE
4150 IF ITT=9 THEN 100
4160 CALL CLEAR
4170 ZEI$="AUF WIEDERSEHEN"
4180 U=12
4190 Z=8
4200 GOSUB 2670
4210 FOR T=1 TO 2000
4220 NEXT T
4230 CALL CLEAR
4240 END
```

LISTINGS

Desperator

Ein ähnliches Spiel hatten wir bereits vor einiger Zeit veröffentlicht. Daß wir unserem Grundsatz untreu wurden, von einem Spiel jeweils auch nur eine Version abzudrucken, hat dieses Programm einzig und alleine seiner guten grafischen Darstellung zu verdanken.
Worum geht es nun bei diesem Spiel?
In einem quadratischen oder rechteckigen Feld, dessen Form und Größe vom Spieler bestimmt wird (zwischen 3x3 und 5x5) sind bis zu 24 Zahlen etwas durcheinandergeraten. Ein Feld bleibt jeweils leer. Mit Hilfe des Joysticks kann der Spieler nun die Position des Leerfeldes mit einer benachbarten Zahl tauschen. Durch diese Tauschmanöver sollen die Zahlen wieder in der richtigen Reihenfolge von 1 bis X angeordnet werden, wobei sich die 1 in der linken oberen Ecke und die letzte Zahl in der rechten unteren Ecke befinden muß. Die ersten Züge sind noch sehr einfach. Dennoch sollte man sich schon früh genug überlegen, wie die letzte Reihe plaziert werden kann, ohne die mühsam angeordneten oberen Reihen wieder komplett durcheinanderzubringen. Da sich die Punktzahl aus der Anzahl der Züge und der benötigten Zeit errechnet, kommt man durch logisches Überlegen weiter als durch sinnloses Herumprobieren. Obwohl dieser Weg natürlich auch zum Ziel führen kann.
Ist das Prinzip dieses Spiels noch nicht bekannt, sollte man zum Anfang mit dem kleinstmöglichen Feld (3x3) beginnen und sich langsam bis zur Professionalität steigern. Natürlich gehört zu diesem Spiel auch eine gehörige Portion Geduld und viel Muse zum Kombinieren.

LISTINGS

```
100 ! ======================
110 ! =                    =
120 ! =    DESPERATION     =
130 ! =    -----------     =
140 ! =                    =
150 ! =  COPYRIGHT '86 BY  =
160 ! =                    =
170 ! = FRANK SCHWEISFURTH =
180 ! =                    =
190 ! =   KONFIGURATION:   =
200 ! =  TI 99/4A KONSOLE  =
210 ! =     EXT. BASIC     =
220 ! =   OPT. JOYST.1/2   =
230 ! =                    =
240 ! =                    =
250 ! =  SPEICHERBELEGUNG  =
260 ! = 11558 Bytes (o.V.) =
270 ! =                    =
280 ! ======================
290 !
3ØØ ON WARNING NEXT :: DIM N
(7,5):: CALL GCI :: RANDOMIZ
E :: CALL GME(N(,))
31Ø FOR I=1 TO 1Ø :: CALL FN
D(N(,),FX,FY,7,5):: CALL INP
(MNX,MNY,FX,FY,ZG,7,5):: CAL
L MOV(N(,),MNX,MNY,FX,FY,ZG)
:: NEXT I
32Ø SUB GCI
33Ø CALL CLEAR :: CALL SCREE
N(7):: RESTORE 35Ø :: CALL M
AGNIFY(3):: CALL COLOR(1,7,6
)
34Ø FOR C=32 TO 143 :: READ
C$ :: CALL CHAR(C,C$):: NEXT
 C :: FOR C=2 TO 14 :: READ
C1,C2 :: CALL COLOR(C,C1,C2)
:: NEXT C :: SUBEXIT
35Ø DATA FFFFFFFFFFFFFFFF,18
183CFFFF3C1818,ØØØØØØFFFF3C1
818,18183CFFFFØØØØØØ,18181C1
F1F1C1818,181838F8F8381818,Ø
ØØØØØ1F1F1C1818
36Ø DATA 18181C1F1FØØØØØØ,ØØ
ØØØØF8F8381818,181838F8F8ØØØ
ØØØ,1818181818181818,ØØØØØØF
FFFØØØØØØ,,,,,ØØ7C868A92A2C2
7C,ØØØ8182848Ø8Ø81C
37Ø DATA ØØ7C82Ø23C4Ø82FE,ØØ
7C82Ø23CØ2827C,ØØ9C8888FEØ8Ø
81C,ØØFE828ØFCØ2827C,ØØ7C828
ØFC82827C,ØØFE84Ø87Ø384Ø8Ø,Ø
Ø7C82827C82827C
38Ø DATA ØØ7C8282FEØ2827C,Ø8
141414Ø8ØØØ8,ØØ7C82420C1ØØ
Ø1Ø,ØØØØØØØØ1818102Ø,ØØØØØØØ
ØØØØØ1818,ØØØØØØ1818ØØ1818,Ø
ØØØ18ØØ1818102Ø
39Ø DATA 3C4299A1A199423C,ØØ
3844447C4444EE,ØØ7C42427C424
2FC,ØØ7C828Ø8Ø8Ø827C,ØØFC424
2424242FC,ØØFE424Ø7Ø4Ø42FE,Ø
ØFE424Ø7Ø4Ø4ØEØ
4ØØ DATA ØØ7C828Ø8C92827C,ØØ
EE44447C4444EE,ØØ381Ø1Ø1Ø1Ø1
Ø38,ØØ381Ø1Ø1Ø5Ø2Ø,ØØE2444
87Ø4844EE,ØØEØ4Ø4Ø4Ø4242FE,Ø
Ø446C5444444444EE
41Ø DATA ØØCE6464544C4CE6,ØØ
7C828282827C,ØØFC42427C4Ø4
ØEØ,ØØ7C8282928C8C72,ØØFC424
27C4844E6,ØØ7C828Ø7CØ2827C,Ø
ØFE921Ø1Ø1Ø1Ø38
42Ø DATA ØØEE444444444438,ØØ
EE4444442828 1Ø,ØØEE444444546
C44,ØØØØØØØØØØØØØØFF,FFØØØØØ
ØØØØØØØØØ,Ø1Ø1Ø1Ø1Ø1Ø1Ø1Ø1,8
Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
43Ø DATA Ø1Ø3Ø7ØF1F3F7FFF,FF
FEFCF8FØEØCØ8Ø,FFFFFFFFFFFFF
FFF,FEFDFBF7EFDFBF7F,7EFFC38
199999999,9999999981C3FF7E,7
EFFF3E3C393F3F3
44Ø DATA F3F3F3F3F3E1FF7E,7E
FFC38199F9F9F3,E7CF9F9F8181F
F7E,7EFFC38199F9F9C3,C3F9F99
981C3FF7E,7EFF9F9F93939381,8
1F3F3F3F3E1FF7E
45Ø DATA 7EFF81819F9F9F83,81
F9F99981C3FF7E,7EFFC3819F9F9
F83,8199999981C3FF7E,7EFF818
1F9F9F3C1,C1E7CFCF9F9FFF7E,7
EFFC381999999C3
46Ø DATA C399999981C3FF7E,7E
FFC38199999981,C1F9F9F981C3F
F7E,7EFFFEFCF881C2EA,C281F8F
CFEFFFF7E,7EFF7F3F1F814357,4
3811F3F7FFFFF7E
47Ø DATA 7FFEFCF8FØEØCØ8Ø,8Ø
CØEØFØF8FCFE7F,FEFFFFFFFFØ3Ø
3Ø3,Ø3Ø3Ø3FFFFFFFFFE,7FFFFFF
FFFCØCØCØ,CØCØCØFFFFFFFF7F,F
E7F3F1FØFØ7Ø3Ø1
48Ø DATA Ø1Ø3Ø7ØF1F3F7FFE,7F
FEFCF8FØEØCØ8Ø,F8F8F8F8F8F8F
F7F,FE7F3F1FØFØ7Ø3Ø1,1F1F1F1
F1F1FFFFE,7FFFF8F8F8F8F8F8,8
ØCØEØFØF8FCFE7F
49Ø DATA FEFF1F1F1F1F1F1F,Ø1
Ø3Ø7ØF1F3F7FFE,FFFFFFFØEØEØE
ØEØ,EØEØEØEØFØFFFFFF,FFFFFFØ
FØ7Ø7Ø7Ø7,Ø7Ø7Ø7Ø7ØFFFFFFF,E
ØEØEØEØEØEØEØEØ
5ØØ DATA Ø7Ø7Ø7Ø7Ø7Ø7Ø7Ø7,FF
FFFFØØØØØØØØØØ,ØØØØØØØØØØFFF
FFF
51Ø DATA 7,6,16,7,16,7,16,7,
16,7,16,7,2,7,8,6,8,6,8,6,8,
6,8,6,5,6
52Ø SUBEND
53Ø SUB SCR(HT,VT,MNX,MNY)
```

```
540 CALL CLEAR :: CH=3*HT+1
:: CV=3*VT+1 :: MNX=15-INT(C
H/2):: MXX=MNX+CH-1 :: MNY=1
5-INT(CV/2):: MXY=MNY+CV-1
550 CALL HCHAR(MNY-4,MNX+1,8
8,CH+2):: CALL HCHAR(MNY-3,M
NX,92):: CALL HCHAR(MNY-3,MN
X+1,94,CH+1):: CALL HCHAR(MN
Y-3,MNX+2+CH,95)
560 CALL HCHAR(MNY-2,MNX-1,9
2):: CALL HCHAR(MNY-2,MNX,94
,CH+1):: CALL HCHAR(MNY-2,MN
X+CH+1,95):: CALL VCHAR(MNY-
1,MNX-2,90,CV+2)
570 CALL VCHAR(MNY-1,MXX+2,9
4,CV+1):: CALL VCHAR(MNY-2,M
XX+3,94,CV+1):: CALL VCHAR(M
XY+1,MXX+2,93):: CALL VCHAR(
MXY,MXX+3,93)
580 CALL VCHAR(MNY-3,MXX+4,9
1,CV+2):: CALL HCHAR(MXY+2,M
NX-1,89,CH+2)
590 CALL HCHAR(MNY-1,MNX-1,1
36):: CALL HCHAR(MNY-1,MNX,1
42,CH):: CALL HCHAR(MNY-1,MX
X+1,138):: CALL VCHAR(MNY,MX
X+1,141,CV)
600 CALL HCHAR(MXY+1,MXX+1,1
39):: CALL HCHAR(MXY+1,MNX,1
43,CH):: CALL HCHAR(MXY+1,MN
X-1,137):: CALL VCHAR(MNY,MN
X-1,140,CV)
610 A$="&++"&RPT$(CHR$(34)&"
++",HT-1)&"(" :: B$=RPT$("*t
v",HT)&"*" :: C$=RPT$("*uw",
HT)&"*"
620 D$="$++"&RPT$("!++",HT-1
)&"%" :: E$="'++"&RPT$("#++"
,HT-1)&")" :: MNX=MNX-2
630 DISPLAY AT(MNY,MNX)SIZE(
CH):A$ :: DISPLAY AT(MNY+1,M
NX)SIZE(CH):B$ :: DISPLAY AT
(MNY+2,MNX)SIZE(CH):C$
640 FOR C=2 TO VT :: Z=(C-2)
*3 :: DISPLAY AT(MNY+Z+3,MNX
)SIZE(CH):D$ :: DISPLAY AT(M
NY+Z+4,MNX)SIZE(CH):B$ :: DI
SPLAY AT(MNY+Z+5,MNX)SIZE(CH
):C$ :: NEXT C
650 DISPLAY AT(MXY,MNX)SIZE(
CH):E$ :: SUBEND
660 SUB NRS(N(,),MNX,MNY,HT,
VT)
670 C=0 :: FOR C1=1 TO VT ::
 FOR C2=1 TO HT :: N(C2,C1)=
C :: CALL SO2(C2*10+110,C1*1
0+110):: CALL DSP(MNX,MNY,C2
,C1,C):: C=C+1 :: NEXT C2 ::
 NEXT C1 :: CALL DRC(MNX,MNY
,1,1,0)
680 FOR C0=1 TO HT*VT/2+5
```

LISTINGS

```
690 C1=INT(RND*HT)+1 :: C3=I
NT(RND*HT)+1 :: C2=INT(RND*V
T)+1 :: C4=INT(RND*VT)+1 ::
C=N(C1,C2):: N(C1,C2)=N(C3,C
4):: N(C3,C4)=C
700 CALL SO2(N(C1,C2)*10+110
,N(C3,C4)*10+110)
710 IF N(C1,C2)=0 THEN CALL
DRC(MNX,MNY,C1,C2,0)ELSE CAL
L DSP(MNX,MNY,C1,C2,N(C1,C2)
)
720 IF N(C3,C4)=0 THEN CALL
DRC(MNX,MNY,C3,C4,0)ELSE CAL
L DSP(MNX,MNY,C3,C4,N(C3,C4)
)
730 NEXT C0 :: SUBEND
740 SUB DSP(MNX,MNY,C2,C3,C1
)
750 A=MNY+(C3-1)*3+1 :: B=MN
X+(C2-1)*3+3 :: D1=96+INT(C1
/10)*2 :: D2=96+((C1/10)-INT
(C1/10))*20
760 CALL HCHAR(A,B,D1):: CAL
L HCHAR(A+1,B,D1+1):: CALL H
CHAR(A,B+1,D2):: CALL HCHAR(
A+1,B+1,D2+1)
770 SUBEND
780 SUB DRC(MNX,MNY,FX,FY,PT
)
790 A=MNY+(FY-1)*3+1 :: B=MN
X+(FX-1)*3+3 :: D1=116+PT*4
800 CALL HCHAR(A,B,D1):: CAL
L HCHAR(A+1,B,D1+1):: CALL H
CHAR(A,B+1,D1+2):: CALL HCHA
R(A+1,B+1,D1+3):: SUBEND
810 SUB INP(MNX,MNY,FX,FY,ZG
,HT,VT,PKT):: ZG=0
820 PKT=PKT-5 :: CALL JOYST(
1,X1,Y1):: CALL JOYST(2,X2,Y
2):: CALL KEY(0,K0,S0):: CAL
L KEY(1,K1,S1):: CALL KEY(2,
K2,S2)
830 IF X1=0 AND Y1=0 AND X2=
0 AND Y2=0 AND S0=0 AND S1=0
 AND S2=0 THEN 820
840 IF (X1=-4 OR X2=-4 OR K0
=83 OR K0=115)AND(FX<>HT)THE
N CALL DRC(MNX,MNY,FX,FY,1):
: ZG=1 :: GOTO 820
850 IF (X1=4 OR X2=4 OR K0=6
8 OR K0=100)AND(FX<>1)THEN C
ALL DRC(MNX,MNY,FX,FY,2):: Z
G=2 :: GOTO 820
860 IF (Y1=-4 OR Y2=-4 OR K0
=88 OR K0=120)AND(FY<>1)THEN
 CALL DRC(MNX,MNY,FX,FY,4)::
 ZG=4 :: GOTO 820
870 IF (Y1=4 OR Y2=4 OR K0=6
9 OR K0=101)AND(FY<>VT)THEN
CALL DRC(MNX,MNY,FX,FY,3)::
```

LISTINGS

```
ZG=3 :: GOTO 82Ø
88Ø IF (KØ=13 OR S1=1 OR S2=
1)AND ZG<>Ø THEN PKT=PKT-3Ø
:: SUBEXIT ELSE IF SØ THEN C
ALL DRC(MNX,MNY,FX,FY,Ø):: G
OTO 82Ø
89Ø GOTO 82Ø :: SUBEND
9ØØ SUB FND(N(,),FX,FY,HT,VT
)
91Ø FOR C1=1 TO HT :: FOR C2
=1 TO VT :: IF N(C1,C2)=Ø TH
EN FX=C1 :: FY=C2 :: SUBEXIT
92Ø NEXT C2 :: NEXT C1 :: SU
BEND
93Ø SUB ERS(MNX,MNY,P1,P2)
94Ø PY=MNY+(P2-1)*3+1 :: PX=
MNX+(P1-1)*3+3
95Ø CALL HCHAR(PY,PX,32):: C
ALL HCHAR(PY+1,PX,32):: CALL
 HCHAR(PY+1,PX+1,32):: CALL
HCHAR(PY,PX+1,32):: SUBEND
96Ø SUB SPR(SN,MNX,MNY,P1,P2
)
97Ø PY=MNY+(P2-1)*3+1 :: PX=
MNX+(P1-1)*3+3 :: CALL GCHAR
(PY,PX,GC):: IF GC>115 THEN
1ØØØ
98Ø TG=44 :: CALL CHARPAT(GC
,C$):: CALL CHAR(44,C$):: CA
LL CHARPAT(GC+1,C$):: CALL C
HAR(45,C$)
99Ø CALL GCHAR(PY,PX+1,GC)::
 CALL CHARPAT(GC,C$):: CALL
CHAR(46,C$):: CALL CHARPAT(G
C+1,C$):: CALL CHAR(47,C$)::
 GOTO 1Ø1Ø
1ØØØ TG=116
1Ø1Ø CALL ERS(MNX,MNY,P1,P2)
:: PX=(PX*8)-7 :: PY=(PY*8)-
7 :: CALL SPRITE(#SN,TG,8,PY
,PX):: SUBEND
1Ø2Ø SUB MOV(N(,),MNX,MNY,FX
,FY,ZG)
1Ø3Ø IF ZG=1 THEN FX2=FX+1 :
: FY2=FY ELSE IF ZG=2 THEN F
X2=FX-1 :: FY2=FY ELSE IF ZG
=4 THEN FX2=FX :: FY2=FY-1 E
LSE FX2=FX :: FY2=FY+1
1Ø4Ø IF FX>FX2 THEN DX=-8 EL
SE IF FX=FX2 THEN DX=Ø ELSE
DX=8
1Ø5Ø IF FY>FY2 THEN DY=-8 EL
SE IF FY=FY2 THEN DY=Ø ELSE
DY=8
1Ø6Ø CALL SPR(1,MNX,MNY,FX,F
Y):: CALL SPR(2,MNX,MNY,FX2,
FY2):: FOR I=1 TO 24 STEP 8
:: CALL SO1(ZG)
1Ø7Ø CALL POSITION(#1,P1,P2)
:: CALL LOCATE(#1,P1+DY,P2+D
X):: CALL POSITION(#2,P1,P2)
:: CALL LOCATE(#2,P1-DY,P2-D
X):: NEXT I
1Ø8Ø N(FX,FY)=N(FX2,FY2):: N
(FX2,FY2)=Ø :: CALL DSP(MNX,
MNY,FX,FY,N(FX,FY)):: CALL D
RC(MNX,MNY,FX2,FY2,Ø):: CALL
 DELSPRITE(ALL)
1Ø9Ø SUBEND
11ØØ SUB TTL(HT,VT)
111Ø CALL CLEAR
112Ø DISPLAY AT(2,6):"&+++++
++++++++(" :: DISPLAY AT(3,6
):"*              *" :: DISPL
AY AT(4,6):"* DESPERATION *"
113Ø DISPLAY AT(5,6):"*
      *" :: DISPLAY AT(6,6
):"'+++++++++++++)" :: CALL
SO3 :: DISPLAY AT(1Ø,3):"@ Ø
4 Ø7 1986 BITSOFT"
114Ø CALL HCHAR(8,1,142,32):
: CALL HCHAR(9,1,143,32):: C
ALL HCHAR(11,1,142,32):: CAL
L HCHAR(12,1,143,32)
115Ø CALL HCHAR(15,1,142,32)
:: CALL HCHAR(16,1,143,32)::
 DISPLAY AT(17,1):"GEWUENSCH
TE FELDHOEHE          5"
116Ø DISPLAY AT(19,1):"GEWUE
NSCHTE FELDBREITE        5" ::
CALL HCHAR(2Ø,1,142,32):: CA
LL HCHAR(21,1,143,32):: ACCE
PT AT(17,28)SIZE(-1)VALIDATE
("345")BEEP:VT
117Ø ACCEPT AT(19,28)SIZE(-1
)VALIDATE("34567")BEEP:HT ::
 DISPLAY AT(24,1):"STARTEN +
+++++++++ KORREKTUR"
118Ø CALL KEY(Ø,K,S):: IF K=
83 OR K=115 THEN SUBEXIT ELS
E IF K=75 OR K=1Ø7 THEN 115Ø
 ELSE 118Ø
119Ø SUBEND
12ØØ SUB CHC(N(,),HT,VT,AR)
121Ø AR=1 :: FOR C1=1 TO VT
:: FOR C2=1 TO HT :: IF C1<>
VT OR C2<>HT THEN IF N(C2,C1
)<>(C1-1)*HT+C2 THEN AR=Ø ::
 GOTO 123Ø
122Ø NEXT C2 :: NEXT C1 :: S
UBEXIT
123Ø AR=2 :: FOR C1=1 TO HT
:: FOR C2=1 TO VT :: IF C1<>
HT OR C2<>VT THEN IF N(C1,C2
)<>(C1-1)*VT+C2 THEN AR=Ø ::
 SUBEXIT
124Ø NEXT C2 :: NEXT C1 :: S
UBEND
125Ø SUB GME(N(,))
126Ø CALL TTL(HT,VT):: CALL
SCR(HT,VT,MNX,MNY):: CALL NR
```

```
S(N(,),MNX,MNY,HT,VT):: PKT=
HT*VT*1ØØØ :: ZA=Ø
127Ø IF PKT<Ø THEN 13ØØ ELSE
 CALL CRE(PKT,ZA,HI):: CALL
FND(N(,),FX,FY,HT,VT)
128Ø ZA=ZA+1 :: CALL INP(MNX
,MNY,FX,FY,ZG,HT,VT,PKT):: C
ALL MOV(N(,),MNX,MNY,FX,FY,Z
G):: CALL CHC(N(,),HT,VT,AR)
:: IF AR=Ø THEN 127Ø
129Ø CALL WIN(ANS,PKT,HI)::
IF ANS=1 THEN 126Ø ELSE CALL
 CLEAR :: END
13ØØ CALL LST(ANS):: IF ANS=
1 THEN 126Ø ELSE CALL CLEAR
:: END
131Ø SUBEND
132Ø SUB CRE(PKT,ZA,HI)
133Ø IF PKT>HI THEN HSC=PKT
ELSE HSC=HI
134Ø DISPLAY AT(1,1):USING "
PKT ##### VER #### HSC #####
":PKT,ZA,HSC :: SUBEND
135Ø SUB WIN(ANS,PKT,HI)
136Ø DISPLAY AT(1,1):USING "
GESCHAFFT !        PKT #####
":PKT :: CALL SO3 :: IF PKT>
HI THEN HI=PKT :: CALL SO3
137Ø DISPLAY AT(2,1):"NOCHMA
L              J ODER N" :: C
ALL KEY(Ø,K,S):: IF K=74 OR
K=1Ø6 THEN ANS=1 ELSE IF K=7
8 OR K=11Ø THEN ANS=Ø ELSE 1
37Ø
138Ø SUBEND
139Ø SUB LST(ANS)
14ØØ DISPLAY AT(1,1):"LEIDER
 VERLOREN !" :: DISPLAY AT(2
,1):"NOCHMAL            J O
DER N" :: CALL SO4
141Ø CALL KEY(Ø,K,S):: IF K=
74 OR K=1Ø6 THEN ANS=1 ELSE
IF K=78 OR K=11Ø THEN ANS=Ø
ELSE 141Ø
142Ø SUBEND
143Ø SUB SO1(A):: CALL SOUND
(-5ØØ,25Ø*A,1Ø,4ØØØØ,3Ø,6ØØ+
A*1ØØ,3Ø,-8,Ø):: SUBEND
144Ø SUB SO2(A1,A2):: CALL S
OUND(1,A1+1,Ø,A1,Ø,A1+2,Ø)::
 CALL SOUND(1,A2+1,Ø,A2,Ø,A2
+2,Ø):: SUBEND
145Ø SUB SO3
146Ø CALL SOUND(1ØØ,22Ø,Ø,22
1,5,222,5):: CALL SOUND(1ØØ,
262,Ø,263,5,264,5):: CALL SO
UND(1ØØ,294,Ø,295,5,296,5)
147Ø CALL SOUND(1ØØ,277,Ø,27
8,5,279,5):: CALL SOUND(1ØØ,
311,Ø,312,5,313,5):: CALL SO
UND(3ØØ,311,Ø,277,Ø,37Ø,Ø)
148Ø SUBEND
149Ø SUB SO4
15ØØ CALL SOUND(1ØØ,147,Ø,14
8,5,149,5):: CALL SOUND(5Ø,1
1Ø,Ø,111,Ø,112,Ø):: CALL SOU
ND(1ØØ,131,Ø,132,5,133,5)
151Ø CALL SOUND(5Ø,11Ø,Ø,111
,Ø,112,Ø):: CALL SOUND(1ØØ,1
23,Ø,124,5,125,5):: CALL SOU
ND(5Ø,11Ø,Ø,111,Ø,112,Ø)
152Ø CALL SOUND(5Ø,11Ø,Ø,117
,Ø,123,Ø):: CALL SOUND(15Ø,1
1Ø,Ø,117,Ø,123,Ø):: SUBEND
```

LISTINGS

Zwei Weg Freqenz Weiche

Wilfried Hartmann hat dieses Programm für Bastler und Lötkolbenfreaks geschrieben, die sich dank ihrer Kenntnisse und Fingerfertigkeiten so manche Mark einsparen möchten.
Hauptsächlich sind mit diesem Hilfsprogramm die HiFi-Fans angesprochen, die sich ihre Frequenzweiche selbst bauen wollen.
Der Preisunterschied zwischen der selbstgebauten und der gekauften, fertig bestückten und festgelegten Trennfrequenz liegt etwa bei 1:3.
Interessant ist dieses Programm vor allem für Kfz-Lautsprecher, da es auf dem Markt zwar einige gute Weichen zu kaufen gibt, für die dann allerdings auch zwischen 100,– und 200,– Mark auf den Tisch gelegt werden müssen.
Da jeder Lautsprecher einen anderen Frequenzgang (Bandbreite) benutzt, kann man mit diesem Programm auf die individuellen Charakteristika der LS eingehen.
Für gehobene Ansprüche (HiFi) dürften besonders die Weichen der 3. und 4. Ordnung interessant sein. TI-User, die über ein Hardcopyprogramm verfügen (z.B. die eingebaute Routine des EX-BASIC II Plus) können sich den Schaltplan gleich Schwarz auf Weiß ausdrucken lassen.

```
110 !**********************
120 !*                    *
130 !*     ZWEI-WEG       *
135 !*   FREQUENZWEICHE   *
140 !*                    *
150 !*    Copyright by    *
160 !*                    *
170 !*  WILFRIED HARTMANN *
180 !*                    *
220 !* Benoetigte Geraete *
230 !*  TI99/4A  Konsole  *
240 !*     EXT. BASIC     *
250 !*                    *
270 !*  SPEICHERBELEGUNG  *
290 !*     11051 Bytes    *
300 !*                    *
310 !**********************
320 !
325 CALL CLEAR :: CALL SCREE
N(16)
330 DISPLAY AT(24,1):"TASTE
DRUECKEN = RUN"
340 CALL KEY(Ø,K,S):: IF S=Ø
 THEN 340
350 CALL CLEAR
360 DISPLAY AT(1,4):"ZWEI-WE
G FREQUENZWEICHE"
370 DISPLAY AT(4,1):"DAS VOR
LIEGENDE PROGRAMM": :"ERLAUB
T EINE FREQUENZWEICHE": :"SE
LBER ZU BAUEN. VON GROSSEM"
380 DISPLAY AT(10,1):"INTERE
SSE IST DABEI, DAS DIE": :"W
EICHE EINE SEHR ZUFRIEDEN":
:"STELLENDE AUFGABE IM Kfz"
390 DISPLAY AT(16,1):"ERFUEL
T. DIE AUF DEM MARKT": :"ERH
AELTLICHEN SIND IN DER": :"R
EGEL VIEL ZU TEUER UND": :"L
EISTEN SOGAR WENIGER ALS"
400 DISPLAY AT(24,1):"WEICHE
N DER 3./4.ORDNUNG. =>"
410 CALL KEY(Ø,K,S):: IF S=Ø
 THEN 410 :: CALL CLEAR
420 DISPLAY AT(1,1):"DER PRE
ISUNTERSCHIED IST IN": :"DER
 REGEL ETWA 1/3 ZU DEN": :"A
UF DEM MARKT BEFINDLICHEN."
430 DISPLAY AT(7,1):"FUER GE
HOBENE ANSPRUECHE,": :"ALSO
ALS Hi-Fi, KOMMT NUR": :"DIE
 WEICHE DER 3. ODER DER": :"
4. ORDNUNG IN FRAGE."
440 DISPLAY AT(15,1):"DIE FL
ANKENSTEILHEIT UND DIE": :"D
AEMPFUNG LIEGEN IM HOER-": :
"BAREN BEREICH. DA DER ZU BE
-": :"STRAHLENDE RAUM IM Kfz
"
450 DISPLAY AT(23,1):"BEGREN
ZT IST, REICHT DORT =>"
460 CALL KEY(Ø,K,S):: IF S=Ø
 THEN 460 :: CALL CLEAR
470 DISPLAY AT(1,1):"EINE WE
ICH DER 2. ODER GAR": :"DER
3.ORDNUNG."
480 DISPLAY AT(6,1):"ENTSPRE
CHEND DER STEILHEIT":"IHRES
KENNLINIENVERLAUFS IM":"UEBE
RNAHMEBEREICH UNTER-"
490 DISPLAY AT(9,1):"SCHEIDE
T MAN WEICHEN MIT:"
500 DISPLAY AT(11,1):" 6dB S
PANNUNGSABFALL/OKTAVE":"
 (1ter ORDNUNG)"
510 DISPLAY AT(14,1):"12dB S
PANNUNGSABFALL/OKTAVE":"
 (2ter ORDNUNG)"
520 DISPLAY AT(17,1):"18dB S
PANNUNGSABFALL/OKTAVE":"
 (3ter ORDNUNG)"
530 DISPLAY AT(20,1):"24dB S
PANNUNGSABFALL/OKTAVE":"
 (4ter ORDNUNG)"
540 DISPLAY AT(24,1):"TASTE
= DATAENEINGABE"
550 CALL KEY(Ø,K,S):: IF S=Ø
 THEN 550
560 CALL CLEAR
570 DATA 129,8Ø8Ø8Ø8Ø8Ø8Ø8ØF
F,13Ø,ØØØØØØØØØØØØØØFF,131,8
Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø,132,8Ø4Ø2Ø1Ø
Ø8Ø4Ø2Ø1,133,Ø1Ø2Ø4Ø81Ø2Ø4Ø8
Ø
580 DATA 134,FFFFFFFFFFFFFFF
F,139,8Ø8ØFFØØØØFF8Ø8Ø,136,2
4242424242424E7,137,ØØFFØØØØ
FFFFFFFF,138,F2F2F2F2F2F2F2F
2,135,Ø1Ø1Ø1Ø1Ø1Ø1Ø1FF
590 DATA 117,ØØØØ242424 3C4Ø8
Ø,1Ø9,ØØØØØØ6C929292ØØ,123,Ø
Ø18187E7E1818ØØ,45,ØØØØØØ7E7
EØØØØØØ
600 FOR I=1 TO 15 :: READ A,
A$ :: CALL CHAR(A,A$):: NEXT
 I
610 CALL SCREEN(15)
620 FOR I=1 TO 11 :: CALL CO
LOR(I,2,1):: NEXT I
630 CALL COLOR(12,7,1,13,5,1
,14,9,1)
640 CALL TITEL
650 DISPLAY AT(15,4):"ZWEI-W
EG-FREQUENZWEICHE"
660 DISPLAY AT(17,2):"1. 1te
r ORDNUNG     6 dB"
670 DISPLAY AT(18,2):"2. 2te
r ORDNUNG    12 dB"
680 DISPLAY AT(19,2):"3. 3te
r ORDNUNG    18 dB"
690 DISPLAY AT(20,2):"4. 4te
```

```
r ORDNUNG    24 dB"
700 DISPLAY AT(21,2):"5. END
E"
710 DISPLAY AT(24,2):"BITTE
WAHL TREFFEN"
720 ACCEPT AT(24,21)BEEP VAL
IDATE("12345")SIZE(1):K
730 GOSUB 2160
740 ON K GOTO 750,930,1270,1
670,2150
750 GOSUB 2170
760 REM
770 LL=Z/(2*PI*FC)*1000
780 CC=1/(2*PI*Z*FC)*1000000
790 DISPLAY AT(18,2):"L1 IN
mH = " :: DISPLAY AT(20,2):"
C1 IN uF = "
800 DISPLAY AT(18,14):USING
"###.##":LL
810 DISPLAY AT(20,14):USING
"###.##":CC
820 DISPLAY AT(2,11)SIZE(2):
"L1" :: CALL HCHAR(1,14,137)
830 DISPLAY AT(9,11)SIZE(2):
"C1" :: CALL HCHAR(8,14,136)
840 CALL HCHAR(10,26,123)::
CALL HCHAR(12,26,45)
850 DISPLAY AT(24,2):"WEITER
= TASTE DRUECKEN"
860 CALL KEY(0,A,S):: IF S=0
 THEN 860
870 GOSUB 2160
880 DISPLAY AT(15,1):"DIE WE
ICHE 1.ORDNUNG -6dB-":"IST D
IE EINFACHSTE IN DER":"PRAXI
S VORKOMMENDE."
890 DISPLAY AT(18,1):"SIE WI
RD FAST NUR IM KfZ":"VERWEND
UNG FINDEN, WEIL":"DER TIEFT
OENER ZUVIELE":"HOHE UND DER
 HOCHTOENER":"ZUVIELE TIEFE
TOENE BEKOMMT"
900 DISPLAY AT(24,1):"TASTE
= ZURUECK ZUM MENUE"
910 CALL KEY(0,K,S):: IF S=0
 THEN 910
920 GOTO 640
930 GOSUB 2170
940 REM
950 LL=SQR(2)*Z/(2*PI*FC)*10
00
960 CC=SQR(2)/(4*PI*FC*Z)*10
00000
970 DISPLAY AT(18,2):"L1+L2
IN mH"
980 DISPLAY AT(20,2):"C1+C2
IN uF"
990 DISPLAY AT(18,14):USING
"###.##":LL
1000 DISPLAY AT(20,14):USING
 "###.##":CC
```

LISTINGS

```
1010 DISPLAY AT(2,11)SIZE(2)
:"L1" :: DISPLAY AT(5,14)SIZ
E(2):"C1"
1020 DISPLAY AT(9,11)SIZE(2)
:"C2" :: DISPLAY AT(12,14)SI
ZE(2):"L2"
1030 CALL HCHAR(1,14,137)::
CALL HCHAR(8,14,136)
1040 CALL VCHAR(2,18,131,4):
: CALL HCHAR(6,18,129):: CAL
L HCHAR(5,18,139)
1050 CALL VCHAR(9,18,131,4):
: CALL HCHAR(13,18,129):: CA
LL HCHAR(12,18,138)
1060 CALL HCHAR(10,26,45)::
CALL HCHAR(12,26,123)
1070 DISPLAY AT(22,1):"{ AUF
 POLARITAET ACHTEN {"
1080 DISPLAY AT(24,2):"WEITE
R = TASTE DRUECKEN"
1090 CALL KEY(0,K,S):: IF S=
0 THEN 1090
1100 GOSUB 2160
1110 DISPLAY AT(15,1):"DIE W
EICHE 2.ORDNUNG -12dB-":"LIE
FERT EIN SYMMETRISCHES":"STR
AHLUNGSDIAGRAMM, DAS IN":"DE
R AKUSTISCHEN BEZUGS-":"EBEN
E DER LS-KOMBINATION"
1120 DISPLAY AT(20,1):"FREQU
ENZUNABHAENGIG IST."
1130 DISPLAY AT(24,1):"TASTE
 = WEITER"
1140 CALL KEY(0,K,S):: IF S=
0 THEN 1140
1150 GOSUB 2160
1160 DISPLAY AT(15,1):"HOCHT
OENER(HT) UND TIEF-":"TOENER
(TT) MUESSEN GEGEN-":"PHASIG
 GESCHALTET SEIN."
1170 DISPLAY AT(18,1):"WERDE
N DIE LS GLEICHPHASIG":"GESC
HALTET, HEBEN SICH DIE":"AUS
GANGSSIGNALE BEI":"DER UEBER
NAHMEFREQUENZ AUF."
1180 DISPLAY AT(22,1):"ES EN
TSTEHT EIN =LOCH=!!"
1190 DISPLAY AT(24,1):"TASTE
 = WEITER"
1200 CALL KEY(0,K,S):: IF S=
0 THEN 1200
1210 GOSUB 2160
1220 DISPLAY AT(15,1):"FUER
HOHE FORDERUNGEN AN":"DIE QU
ALITAET DER LAUT-":"SPRECHER
WIDERGABE IST EIN":"SPANNUNG
SABFALL VON 12dB"
1230 DISPLAY AT(19,1):"NOCH
ZU KLEIN. SIE REICHT":"ALLER
DINGS DURCHAUS FUER":"ANSPRU
```

```
ECHE IM KfZ."
1240 DISPLAY AT(24,1):"TASTE
 = ZURUECK ZUM MENUE"
1250 CALL KEY(0,K,S):: IF S=
0 THEN 1250
1260 GOTO 640
1270 GOSUB 2170
1280 DISPLAY AT(15,10)SIZE(2
):"mH" :: DISPLAY AT(15,23)S
IZE(2):"uF"
1290 DISPLAY AT(17,1):"L1":"
L2":"L3"
1300 LL1=3*Z/(4*PI*FC)*1000
1310 DISPLAY AT(17,7):USING
"###.##":LL1
1320 LL2=Z/(4*PI*FC)*1000
1330 DISPLAY AT(18,7):USING
"###.##":LL2
1340 LL3=3*Z/(8*PI*FC)*1000
1350 DISPLAY AT(19,7):USING
"###.##":LL3
1360 CC1=2/(3*PI*FC*Z)*10000
00
1370 DISPLAY AT(17,20):USING
 "###.##":CC1
1380 CC2=1/(3*PI*FC*Z)*10000
00
1390 DISPLAY AT(18,20):USING
 "###.##":CC2
1400 CC3=1/(PI*FC*Z)*1000000
1410 DISPLAY AT(19,20):USING
 "###.##":CC3
1420 DISPLAY AT(17,15)SIZE(2
):"C1" :: DISPLAY AT(18,15)S
IZE(2):"C2" :: DISPLAY AT(19
,15)SIZE(2):"C3"
1430 DISPLAY AT(2,11)SIZE(2)
:"L1" :: DISPLAY AT(5,14)SIZ
E(2):"C1" :: DISPLAY AT(2,18
)SIZE(2):"L2"
1440 DISPLAY AT(12,14)SIZE(2
):"L3" :: DISPLAY AT(9,11)SI
ZE(2):"C2" :: DISPLAY AT(9,1
8)SIZE(2):"C3"
1450 CALL HCHAR(1,14,137)::
CALL HCHAR(1,21,137):: CALL
HCHAR(8,14,136):: CALL HCHAR
(8,21,136)
1460 CALL VCHAR(2,18,131,4):
: CALL HCHAR(6,18,129):: CAL
L HCHAR(5,18,139)
1470 CALL VCHAR(9,18,131,4):
: CALL HCHAR(13,18,129):: CA
LL HCHAR(12,18,138)
1480 DISPLAY AT(22,3):"{ AUF
 POLARITAET ACHTEN {"
1490 CALL HCHAR(10,26,123)::
 CALL HCHAR(12,26,45)
1500 DISPLAY AT(24,1)BEEP:"W
EITER = TASTE DRUECKEN"
1510 CALL KEY(0,K,S):: IF S=
0 THEN 1510
1520 GOSUB 2160
1530 DISPLAY AT(15,1):"WEICH
EN 3.ORDNUNG WEISEN":" EINE
FLANKENSTEILHEIT":"IHRER DAE
MPFUNG VON":"18dB/OKTAVE AUF
."
1540 DISPLAY AT(24,1):"TASTE
 = WEITER"
1550 CALL KEY(0,K,S):: IF S=
0 THEN 1550
1560 GOSUB 2160
1570 DISPLAY AT(15,1):"DAS B
IETET GEWAEHR, DASS":"DEN AN
GESCHLOSSENEN LS NUR":"DIE F
REQUENZEN ZUGEFUEHRT":"WERDE
N, WO SIE OPTIMAL"
1580 DISPLAY AT(19,1):"GUENS
TIGE UEBERTRAGUNGS-":"EIGENS
CHAFTEN AUFWEISEN."
1590 DISPLAY AT(24,1):"TASTE
 = WEITER"
1600 CALL KEY(0,K,S):: IF S=
0 THEN 1600
1610 GOSUB 2160
1620 DISPLAY AT(15,1):"WERDE
N DIE LS GEGENPHASIG":"GESCH
ALTET, IST AUCH DIE":"PHASEN
VERZERRUNG GERING !"
1630 DISPLAY AT(19,1):"NACHT
EIL: DAS RICHTDIAGRAMM":"DER
 LS-KOMBINATIONN IST":"NICHT
 SYMMETRISCH AUF DIE":"LS-BE
ZUGSACHSE"
1640 DISPLAY AT(24,1):"TASTE
 = ZURUECK ZUM MENUE"
1650 CALL KEY(0,K,S):: IF S=
0 THEN 1650
1660 GOTO 640
1670 GOSUB 2170
1680 DISPLAY AT(15,9):"mH
  uF"
1690 DISPLAY AT(17,1):"L1":"
L2":"L3":"L4"
1700 LL1=1.8856*Z/(2*PI*FC)*
1000
1710 DISPLAY AT(17,6):USING
"###.##":LL1
1720 LL2=.94281*Z/(2*PI*FC)*
1000
1730 DISPLAY AT(18,6):USING
"###.##":LL2
1740 LL3=Z/(2*PI*FC*1.5910)*
1000
1750 DISPLAY AT(19,6):USING
"###.##":LL3
1760 LL4=Z/(2*PI*FC*.35355)*
1000
1770 DISPLAY AT(20,6):USING
"###.##":LL4
```

LISTINGS

LISTINGS

```
178Ø CC1=1.591Ø/(2*PI*FC*Z)*
1ØØØØØØ
179Ø DISPLAY AT(17,2Ø):USING
 "###.##":CC1
18ØØ DISPLAY AT(17,15)SIZE(2
):"C1"
181Ø CC2=.35355/(2*PI*FC*Z)*
1ØØØØØØ
182Ø DISPLAY AT(18,2Ø):USING
 "###.##":CC2
183Ø DISPLAY AT(18,15)SIZE(2
):"C2"
184Ø CC3=1/(2*PI*FC*1.8856*Z
)*1ØØØØØØ
185Ø DISPLAY AT(19,2Ø):USING
 "###.##":CC3
186Ø DISPLAY AT(19,15)SIZE(2
):"C3"
187Ø CC4=1/(2*PI*FC*.94281*Z
)*1ØØØØØØ
188Ø DISPLAY AT(2Ø,2Ø):USING
 "###.##":CC4
189Ø DISPLAY AT(2Ø,15)SIZE(2
):"C4"
19ØØ DISPLAY AT(2,11)SIZE(2)
:"L1" :: DISPLAY AT(2,17)SIZ
E(2):"L2" :: DISPLAY AT(5,13
)SIZE(2):"C1" :: DISPLAY AT(
5,18)SIZE(2):"C2"
191Ø DISPLAY AT(12,13)SIZE(2
):"L3" :: DISPLAY AT(12,18)S
IZE(2):"L4" :: DISPLAY AT(9,
11)SIZE(2):"C3" :: DISPLAY A
T(9,17)SIZE(2):"C4"
192Ø FOR I=14 TO 2Ø STEP 6 :
: CALL HCHAR(1,I,137):: CALL
 HCHAR(8,I,136):: NEXT I
193Ø CALL VCHAR(2,17,131,5):
: CALL HCHAR(6,17,129):: CAL
L VCHAR(2,22,131,5):: CALL H
CHAR(6,22,129)
194Ø CALL VCHAR(9,17,131,5):
: CALL HCHAR(13,17,129):: CA
LL VCHAR(9,22,131,5):: CALL
HCHAR(13,22,129)
195Ø FOR I=17 TO 22 STEP 5 :
: CALL HCHAR(5,I,139):: CALL
 HCHAR(12,I,138):: NEXT I
196Ø CALL HCHAR(1Ø,26,45)::
CALL HCHAR(12,26,123)
197Ø DISPLAY AT(22,1):"(( AU
F POLARITAET ACHTEN (("
198Ø DISPLAY AT(24,2):"WEITE
R = TASTE DRUECKEN"
199Ø CALL KEY(Ø,K,S):: IF S=
Ø THEN 199Ø
2ØØØ GOSUB 216Ø
2Ø1Ø DISPLAY AT(15,1):"BUTTE
RWORTH-WEICHEN DER":"4.ORDNU
NG ZEICHNEN SICH,":"ABGESEHE
N VON DER GROSSEN":"FLANKENS
TEILHEIT DER"
2Ø2Ø DISPLAY AT(19,1):"DAEMP
FUNG DADURCH AUS, DASS":"DIE
 ACHSE DER STRAHLUNGS-":"KEN
NLINIE MIT DER LS-BE-":"ZUGS
ACHSE UEBEREINSTIMMT."
2Ø3Ø DISPLAY AT(24,1):"TASTE
 = WEITER"
2Ø4Ø CALL KEY(Ø,K,S):: IF S=
Ø THEN 2Ø4Ø
2Ø5Ø GOSUB 216Ø
2Ø6Ø DISPLAY AT(15,1):"SIE I
ST UNABHAENGIG VON DER":"FRE
QUENZ. TIEFTOENER UND":"HOCH
TOENER MUESSEN VON DER":"GLE
ICHEN BEZUGSACHSE"
2Ø7Ø DISPLAY AT(19,1):"STRAH
LEN. zB.:VERTIKAL":"UEBEREIN
ANDER ODER IN DER":"HORIZONT
ALEBENE GEGENSEITIG":"SO VER
SCHIEBEN, DASS DIE"
2Ø8Ø DISPLAY AT(24,1):"TASTE
 = WEITER"
2Ø9Ø CALL KEY(Ø,K,S):: IF S=
Ø THEN 2Ø9Ø
21ØØ GOSUB 216Ø
211Ø DISPLAY AT(15,1):"AKUST
ISCHEN ZENTREN DER ":"BEIDEN
 LAUTSPRECHER UEBER-":"EINAN
DER LIEGEN."
212Ø DISPLAY AT(24,1):"TASTE
 = ZURUECK ZUM MENUE"
213Ø CALL KEY(Ø,K,S):: IF S=
Ø THEN 213Ø
214Ø GOTO 64Ø
215Ø CALL CLEAR :: END
216Ø CALL HCHAR(15,1,32,32Ø)
:: RETURN
217Ø DISPLAY AT(17,2):"WIDER
STAND DES":"LS IN OHM :"
218Ø ACCEPT AT(18,16)BEEP:Z
219Ø DISPLAY AT(2Ø,2):"WELCH
E TRENN-":"FREQUENZ ? "
22ØØ ACCEPT AT(21,14)SIZE(5)
BEEP:FC
221Ø DISPLAY AT(24,5):"RICHT
IG ? J/N"
222Ø CALL KEY(Ø,A,S):: IF S=
Ø THEN 222Ø :: IF A=74 THEN
223Ø :: IF A=78 THEN 218Ø
223Ø GOSUB 216Ø :: RETURN
224Ø SUB TITEL :: CALL CLEAR
225Ø FOR I=7 TO 23 :: CALL H
CHAR(1,I,13Ø):: CALL HCHAR(6
,I,13Ø):: NEXT I
226Ø CALL VCHAR(2,1Ø,13,6)::
 CALL HCHAR(6,11,129)
227Ø CALL HCHAR(8,11,132)::
CALL HCHAR(8,12,13Ø,12):: CA
LL VCHAR(7,1Ø,131,6):: CALL
```

```
HCHAR(13,1Ø,132)
228Ø CALL HCHAR(13,11,13Ø,13
)
229Ø CALL HCHAR(3,26,123)::
CALL HCHAR(4,26,134):: CALL
HCHAR(5,26,45):: CALL HCHAR(
3,27,133):: CALL HCHAR(5,27,
132)
23ØØ FOR I=2 TO 9 STEP 7 ::
CALL VCHAR(I,24,132):: CALL
VCHAR(I+1,25,132)
231Ø CALL VCHAR(I+3,25,133):
: CALL VCHAR(I+4,24,133):: N
EXT I
232Ø CALL HCHAR(11,26,134)::
 CALL HCHAR(1Ø,27,133):: CAL
L HCHAR(12,27,132)
233Ø CALL VCHAR(2,11,131,4):
: CALL HCHAR(7,11,131):: CAL
L HCHAR(6,1Ø,13Ø)
234Ø CALL HCHAR(4,27,84,2)::
 CALL HCHAR(11,27,72):: CALL
 HCHAR(11,28,84)
235Ø CALL HCHAR(1,6,123):: C
ALL HCHAR(6,6,45)
236Ø SUBEND
```

HCA
WIR STEHEN IHNEN MIT RAT UND TAT ZUR SEITE

Dictionary

Dieses Programmpaket besteht aus insgesamt fünf Basic-Programmen, die sich teilweise selbst aufrufen. Mit diesem Programm können Vokabeln sehr gut verwaltet werden, aber durch Änderungen können auch andere Problemlösungen mit diesem Programm bearbeitet werden. Als Speichermedium ist eine Diskettenstation vorgesehen, weil die Vokabeln als Random-Datei abgelegt werden. Außerdem ist der Zugriff auf die einzelnen Programme nicht so gut mit einem Kassettenrecorder zu verwirklichen.

Das Programmpaket umfaßt folgende Programme:

Hauptmod	=	Hauptprogramm (ruft alle anderen Programme auf
Progbmod	=	Programmbeschreibung
Eingabem	=	Eingabeprogramm
Überschm	=	Überschreibeprogramm
Anschaum	=	Anschauungsprogramm
Uebersetzm	=	Übersetzungsprogramm
Loesch	=	Löschprogramm

Alle anderen Programme verzweigen nach ihrer Beendigung wieder in das Hauptprogramm.

Die Programmbeschreibung ist sehr sauber ausgearbeitet und bedarf keiner Ergänzung.

```
10 '****************************** < 58>
11 '*                            * < 58>
12 '*   D I C T I O N A R Y      * < 58>
13 '*       Eingabe-Modul        * < 58>
14 '*                            * < 58>
15 '*                            * < 58>
16 '*     Daniel    Herzog       * < 58>
17 '*     Blumensteinstr. 8      * < 58>
18 '*                            * < 58>
19 '*                            * < 58>
20 '*      CH-3012  Bern         * < 58>
22 '*                            * < 58>
23 '*                            * < 58>
24 '****************************** < 58>
25 '*                            * < 58>
26 '*     M S X - R E V U E      * < 58>
27 '*                            * < 58>
28 '*         1987               * < 58>
29 '*                            * < 58>
30 '****************************** < 58>
31 :                               < 58>
32 :                               < 58>
100 MAXFILES=2:SCREEN0:WIDTH40:K
EYOFF:POKE&HFCAB,&HFF:FORT=2552T
O2559:VPOKET,&H0:NEXT:ON STOP GO
SUB 1110:STOP ON                  <10397>
110 OPEN "A:sprachen.dat" AS #1
LEN=24                            <30453>
120 FIELD #1,12 AS ERSTE$,12 AS
ZWEITE$                           <18579>
130 GET #1,1:CLOSE#1              <54168>
140 GOSUB1040:CLS                 < 5155>
150 TAB1=(40-((LEN(E$)+LEN(Z$)+3
)))                               <15746>
160 IF(INT(TAB1/2))*2<>TAB1THENT
AB1=TAB1-1                        <42753>
170 TAB1=TAB1/2                   <31649>
180 CLS:PRINT"┌──────────────────
──────────────┐│_____________
D_I_C_T_I_O_N_A_R_Y_____________||_
___________________________________
_____|";                          <56615>
190 PRINT"├──────────────────────
─────────────┤|__COPYRIGHT_B
Y_HERZOG_DANIEL_SCHWEIZ__|└───────
──────────────────────────────────
─┘"                               <28928>
200 LOCATETAB1,2:PRINTE$;:PRINT"
_/_";:PRINTZ$                     < 9371>
210 LOCATE0,9:PRINT"_____________
ERFASSUNGSMENU_____________________
_________──────────────────"      <23080>
220 PRINT:PRINT:PRINT"_____1._";
:PRINTE$;:PRINT"_/_";:PRINTZ$     <14559>
230 PRINT"_____2._";:PRINTZ$;:PR
INT"_/_";:PRINTE$                 <63488>
240 PRINT"_____3._ENDE"           <37288>
250 LOCATE0,21:PRINT"____________
____DRÜCKEN_SIE_1,2_ODER_3"       <13431>
260 O$=INKEY$:IFO$="3"THENRUN"HA
```

TI REVUE

Das Magazin für TI 99-4A

Nr. 10/87

Bremen rief und Alle kamen

DSR-Programierung Teil II

32-K-Erweiterung für 25 DM

Das GRAM-Modul im Test

Tips & Tricks

TI 99-4A:

DIALOG

Basic-Programme editieren, Datei verschwunden, Hard-Disk am Ti, und viele weitere Fragen ab Seite 38

TEST & TECHNIK

Erfahrungen mit Turbo-Pasc'99
Keine relativen Dateien möglich ab Seite 41

Gram-Loader-Modul
Eine sinnvolle Erweiterung für Ihren TI ab Seite 42

TIPS & TRICKS

32K-Erweiterung
Rüsten Sie Ihren TI auf! ab Seite 39

Die Lösung zum Schulhaus-Adventure
Wir zeigen Ihnen, wie es weiter-geht . . . ab Seite 51

SERIE & SERVICE

DSR-Programmierung – Teil 2
Fortsetzung der Serie aus dem letzten Heft 6/87 ab Seite 43

Report: TI-Treffen in Bremen
Reportage vor Ort ab Seite 34

LISTINGS

Programm-Lader für Gram-Modul
Dieses Maschinenprogramm ersetzt das Basic-Listing aus Heft 6/87 ab Seite 43

Scrabble 86
Wortspielereien auf dem TI ab Seite 53

RAM-Anzeige
Ein hilfreiches Listing ab Seite 55

Desktop-Fighters
Ein Spiel der Spitzenklasse ab Seite 56

ALLGEMEINES
Börse
Kleinanzeigen unserer Leser ab Seite 62

Das **Impressum** finden Sie auf Seite 52

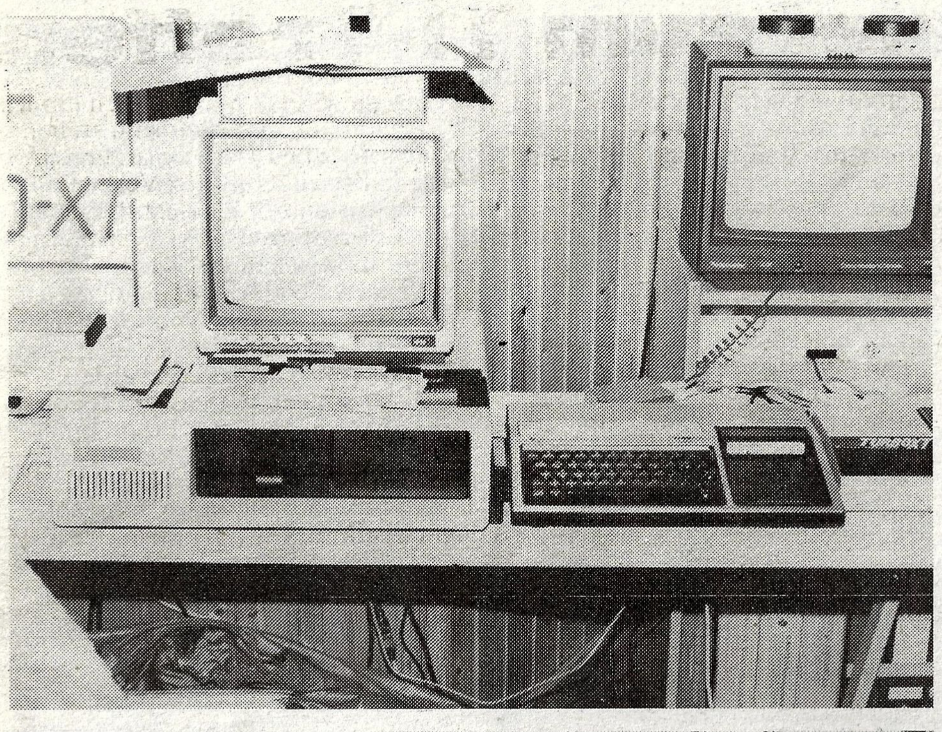
Bremen rief zum TI-Treffen

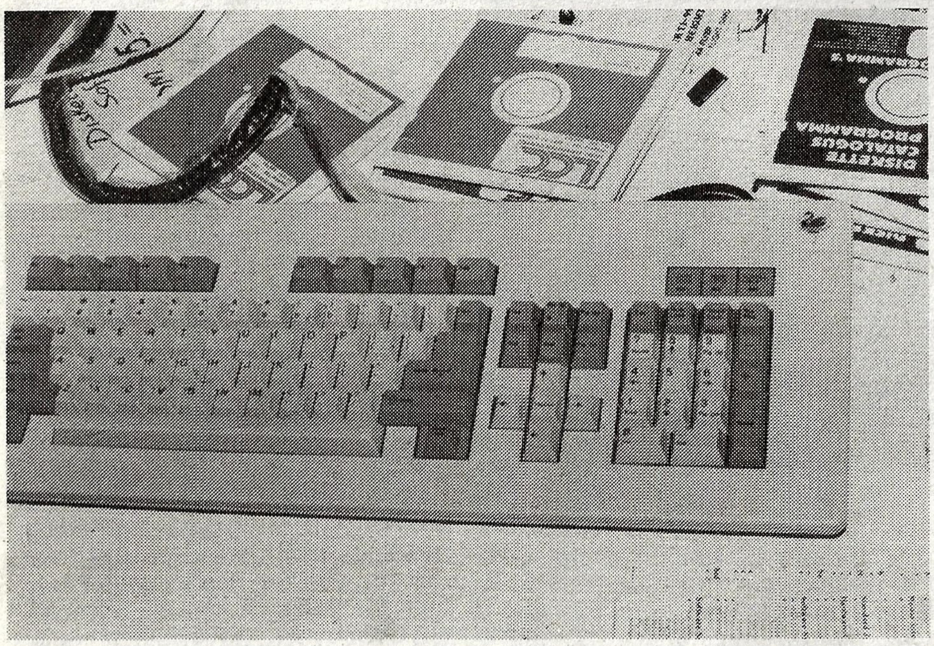
Nach dem gelungenen Treffen in Köln im November 1986 hatte in diesem Jahr die TI-User-Group Bremen die Aufgabe übernommen, diese Veranstaltung zu organisieren. Und trotz der doch allmählich kleiner werdenden Schar der TI99-Anhänger gelang es den Bremern, das Treffen in Köln noch in den Schatten zu stellen. Dies soll auf gar keinen Fall die Leistung des TI-Workshop Rheinland im letzten Jahr schmälern, aber was diesmal im Gemeindezentrum Obervieland den Anhängern des TIs geboten wurde, war absolute Spitzenklasse.

Die ersten Teilnehmer, die außer aus dem Bundesgebiet auch aus Österreich, aus den Niederlanden und aus Belgien angereist waren, trafen bereits am späten Freitag in Bremen ein.
Am Samstag pünktlich um 10 Uhr, begann der offizielle Teil des 2. Internationalen TI-Treffens. Es wurde alles gezeigt, was das Herz eines TI99-Freaks höher schlagen läßt. Hier stellte sich wieder heraus, daß der TI99/4A allen Unkenrufen zum Trotze noch lange nicht tot ist, sondern sich sowohl auf der Softwareseite als auch hardwaremäßig in seinem zweiten Frühling bester Gesundheit erfreut. Viele Dinge, die zu „Texas-Zeiten" unvorstellbar waren, gehören heute zum Standard dieses immer noch faszinierenden Systems. Die Möglichkeiten, die dieser Computer bietet, scheinen trotz seines fast schon „biblischen" Alters unerschöpflich.
Von dem Angebot, die Aktivitäten ihres Clubs auf einer Videowand vorzustellen, machten leider nur wenig Gebrauch. So blieb den Besuchern, die sich informieren wollten, nur der Weg zu den einzelnen Ständen, wo dann allerdings auch detaillierte Fragen ausführlich behandelt werden konnten.
Die 80-Zeichenkarte, die im letzten Jahr in Köln vorgestellt wurde, ist dank der Hilfe des TI-Workshop Rheinland auf dem besten Wege, ihre Kinderkrankheiten abzulegen.
Auch Software, wie zum Beispiel Multiplan und TI-Mac, ein neues Textverarbeitungsprogramm, ist inzwischen für diese nützliche Erweiterung zu bekommen. Besonders beeindruckt hat mich ein Platinen-Entflechtungs-Programm, das ich in dieser hervorragenden Form dem TI gar nicht zugetraut hätte.
Aber wie sagt der Wahlspruch der Rheinländer: „Nichts ist unmöglich". Wollen wir hoffen, daß die Mannen aus Köln und Umgebung auch weiterhin so erfolgreich den Geheimnissen des TI auf der Spur bleiben.
Aber es gibt natürlich noch etliche andere Clubs, die sich dem Innenleben dieses Rechners verschrieben haben. So hat der User-Club Berlin das GROM 0 des TI99/4A von Grund auf überarbeitet. Dabei ist es gelungen, einige Routinen sinnvoll zu kürzen und somit schneller zu machen. Auch haben sie dieses GROM von 6K-Byte auf insgesamt 8KByte erweitert und unter anderem die echten Kleinbuchstaben im Betriebssystem eingebunden. Auch neu aus Berlin ist das Editor/Assembler Modul Version 4, welches ebenfalls wie das Original komplett in GPL programmiert ist. Die dazugehörigen Programme müssen nicht mehr von DSK1 geladen werden, dies ist jetzt von DSK1 bis DSK5 möglich. Sinnvollerweise enthält dieses Modul jetzt auch ein Katalogprogramm. Weiterhin stehen dem Benutzer auch zusätzlich 9 neue Basicbefehle zur Verfügung. Wie man sieht, ist selbst das von Texas Instruments nie preisgegebene Geheimnis des TI99, die Programmiersprache GPL in den GROMs, heute keine Hürde mehr.

Beim 2. Internationalen TI-Treffen in Bremen konnten die interessierten TI-User alles finden, was das Herz begehrt

Ein Beispiel hierfür sind ja auch die GPL-Berichte von Sven Dyroff, die schon seit einiger Zeit in der TI-REVUE erscheinen.
Der Panorama Computer Club aus Duisburg hatte unter anderem eine Modelleisenbahnanlage mitgebracht, auf der, gesteuert durch einen TI99/4A und ein selbstentwickeltes Interface (einer 32 Bit Input/Output-Karte, die auch einzeln angeboten wird), eine Dampflok scheinbar eigene Intelligenz entwickelte. Der hier ausgestellte Prototyp dieser Anlage arbeitete zur besseren Verständlichkeit mit laut klickenden Relais, dasselbe ist aber auch mit Halbleitern machbar, was dann natürlich völlig lautlos funktioniert. Auch eine DSR-Platine, die mit einem batteriegepuffertem 8KByte-RAM das Erstellen und Testen eigener Device-Service-Routinen ermöglicht, war hier zu sehen.
Der TI-Club Baunatal hatte sich Gedanken darüber gemacht, wie man die Kühlung des Netzteiles unter dem Modulschacht des TI99/4A verbessern kann. An dieser Stelle





Zum Tastaturtreiber degradiert: TI-99/4A und IBM XT in wilder Ehe (oben) Geneve-Exote in TI-Kreisen

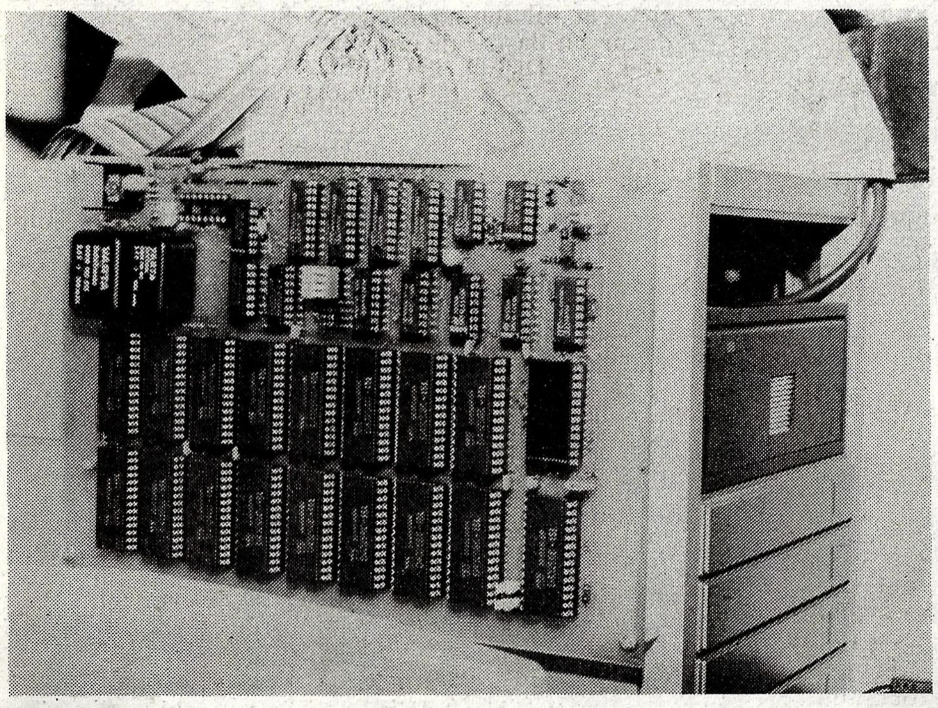
werden die Konsolen nach einiger Zeit doch ganz beachtlich heiß. Und da die Hitzeentwicklung sich auch auf die Lebensdauer der Halbleiterbausteine auswirkt, kann von dieser Idee so mancher TI-User profitieren.
Eine weitere Neuheit in Bremen war die 512KByte Ramdisk des TI-Workshop Wiesbaden. Leider war die Software zu dieser Karte für die Peri-Box des TI99 nicht rechtzeitig fertig geworden, so daß die Möglichkeiten dieser Erweiterung nur teilweise vorgeführt werden konnten. Aber wer sich in der Programmierung von DSR-Routinen auskennt, weiß, wie viel Arbeit darin steckt. Bis das Betriebssystem dieser Karte, das bis zu 32KByte betragen kann, fertig ist, werden noch einige Wochen ins Land gehen. Dann aber soll diese Karte, die zusätzlich noch mit einer Hardware-Uhr ausgestattet ist, allen erdenklichen Komfort bieten. So ist es zum Beispiel heute schon möglich, bis zu 254 Device-Namen selbst zu definieren, wodurch auch nicht vorhandene Peripherie-Geräte simuliert werden können (z.B. TP oder DSK6).

Ebenfalls aus dem Rhein-Main-Gebiet kamen die Mitglieder der System 99 Usergroup. Die TI-Freunde aus Frankfurt zeigten, daß sich der TI99/4A durch eine von 3 auf etwa 3.5 Megahertz erhöhte Taktfrequenz sichtbar schneller machen läßt. Einige hatten ihre Rechner in neue Gehäuse eingebaut und auch mit neuen Tastaturen versehen, so daß der gute alte TI99/4A gar nicht mehr zu erkennen war. Aber auch auf der Softwareseite hatte dieser Club einiges zu bieten. So wurden a. der Flugsimulator SPAD/13-Mark2, ein Mandelbrotmengen-Grafik-Programm sowie ein neues Kopierprogramm vorgeführt.
Die wohl weiteste Anreise hatte ein Mitglied des TI-Clubs Vienna aus Wien. Nachdem er einige Schwierigkeiten hatte, seine Anlage durch den Zoll zu bekommen, konnte er einige sehr beachtenswerte Programme für seine GRAM-Karte vorführen. Leider ist es ja so, daß es mehrere verschiedene Arten von GRAM-Erweiterungen gibt, die nicht miteinander kompatibel sind. Vielleicht sollten sich die Clubs da mal auf einen einheitlichen Standard einigen, um wenigstens die Software hierzu austauschen zu können.

CLUBS AUS ALLER WELT

Nicht unerwähnt bleiben sollen auch die weiteren Clubs, die unter anderem aus Kiel, Essen, München, aber auch aus den Niederlanden und Belgien nach Bremen gekommen waren, um ihre Software- und Hardwareneuheiten auszustellen. Aber auch einige Händler waren vertreten. So konnte man den Nachfolger des TI99, den im letzten Jahr in Köln erstmals vorgestellten Geneve 9640 von Myarc, in aller Ruhe betrachten. Der Rechner besteht aus einer Karte für die Peri-Box des TI99/4A, an die eine IBM-ähnliche PC-Tastatur angeschlossen wird. Der Disk-Controller und die RS232-Karte können wie bisher weiter benutzt werden, die 32KByte-Karte ist überflüssig und muß aus der Box entfernt werden, da der Rechner selbst schon über 512KByte Speicher verfügt. (Diejenigen, die schon bisher ihre 32K-Erweiterung in der TI99/Konsole hatten, haben dann zusammen mit dem Extended Basic Modul eine komplette zweite Anlage, denn auch dieses erweiterte Basic ist im Geneve bereits vorhanden.) Ein Ausbau dieses Rechners bis auf insgesamt 2 Megabyte ist möglich.

Leider ist der Preis mit ca. DM 1200,– für die Grundversion doch ziemlich hoch.
Eine weitere Neuheit, die in Bremen gezeigt wurde, war ein „IBM-kompatibler" TI99/4A. Hier allerdings zeigten sich doch die Grenzen des Texaners, der praktisch nur noch als Tastatur für einen Turbo-XT fungierte. Nach dem Starten des Programms zeigte der TI nur noch einen abgeschalteten Bildschirm. Das Programm lief dann ausschließlich auf dem XT, der auch das dazugehörige Bild auf seinem Monitor lieferte.
Zum Abschluß sollen auch zwei Exoten aus der „TI-Familie" nicht unerwähnt bleiben. Der erste war ein gläserner TI, der am Stand der



Speicher in Reih und Glied: Neue Speichererweiterung für den TI-99/4A

Bremer User-Group zu bewundern war. Der ganze Rechner bestand zwar noch aus den Original-TI-Platinen, war aber in einem durchsichtigen Koffer untergebracht, so daß man das Innenleben in seiner ganzen Schönheit bewundern konnte. Noch einen Schritt weitergegangen war Sven Dyroff vom TI-Workshop Wiesbaden. Dieser hatte einen TI99/4A in wirklich alle Einzelteile zerlegt, nach seinen eigenen Ideen modifiziert und das Ganze dann auf normalen Streifenrasterplatinen wieder aufgebaut, die er dann auf eine Holzplatte geschraubt hatte. Und obwohl dieser Aufbau der Abfallkiste einer Elektronikwerkstatt ähnlicher sah als einem Rechner, geschah das Wunder und der Erbauer konnte seinen TI dem staunenden Publikum in Funktion vorführen.
Aber auch für diejenigen, die nicht nur an den Neuheiten, sondern auch an alt bewährtem Zubehör für ihren TI99/4A interessiert waren, war bestens gesorgt. Der gleichzeitig stattfindende Flohmarkt bot so manchem die Gelegenheit, schon lange gesuchtes Zubehör für seine Anlage günstig zu bekommen. Hier gab es wirklich alles, angefangen von Modulen bis hin zu kompletten Anlagen mit voll ausgebauten Erweiterungsboxen. Auch Diskettenlaufwerke, angefangen von den alten SS/SD TI-Drives mit 90KByte bis hin zur kompletten Doppelstation mit 2 DS/DD-Laufwerken komplett mit Netzteil als DSK2 und DSK3 mit insgesamt bis zu 720KByte konnte man mit nach Hause nehmen. Speichererweiterungen mit 8 oder 16 Bit-Datenbus, RS-232-Karten, Diskcontroller und sonstiges Zubehör war in allen denkbaren Versionen zu bekommen. Auch defektes Zubehör, das für Bastler interessant sein dürfte, war günstig zu haben. So dürfte hier wohl jeder auf seine Kosten gekommen sein.
Wie nicht anders zu erwarten auf einem solchen Treffen, stand der Softwaretausch ganz im Mittelpunkt des Interesses. Die Programmvielfalt für den TI99 ist inzwischen kaum noch zu überblicken, aus diesem Grunde will ich hier auch nicht auf Einzelheiten eingehen. Derjenige, der Infos über bestimmte Programme haben möchte, sollte sich am sinnvollsten mit einem der TI-Clubs in Verbindung setzen. Da es für den TI99 inzwischen ein Riesenangebot an Freeware gibt, läßt sich fast jedes Problem lösen. Doch zum Thema Freeware muß leider auch noch ein ernstes Wort gesagt werden.
In Bremen zeigte sich, daß teilweise die Weitergabe dieser Programme mit dem Kauf einer Diskette zum überhöhten Preis gekoppelt wurde. Man bekam diese Programme also nicht auf eine mitgebrachte Diskette kopiert, sondern mußte eine neue kaufen. Und es widerspricht ja eindeutig dem Freeware-Gedanken, wenn diese Leute mit Programmen, die andere geschrieben haben und den TI-Anhängern kostenlos zur Verfügung stellen wollen, auf diese Art und Weise versuchen, Kasse zu machen. Es ist

GESCHÄFTE MIT FREEWARE

nichts dagegen zu sagen, wenn man seine selbstgeschriebenen Programme verkauft, aber mit Freeware, die andere erdacht haben, ist das nicht in Ordnung. Oder? Euere Meinung dazu würde mich interessieren.
Meldet euch doch mal in der HCA-Mailbox und schreibt eure Meinung. Mich erreicht ihr in dieser Box unter BONGO, ihr könnt eure Meinung aber auch in das Brett 34 schreiben.
Nun wieder zurück zum TI-Treffen. Diejenigen Teilnehmer, die gleich nach dem offiziellen Ende der Veranstaltung um 18 Uhr nach Hause gefahren waren, haben doch einiges verpaßt. Der Höhepunkt des gemütlichen Abends war das kalte Büffet, das von den Frauen der Bremer User-Group vorbereitet worden war. Hier hat es keinen Zweck, etwas zu erzählen, das muß man ganz einfach erlebt haben. Den Dankesworten, die an diesem Abend von Vertretern der einzelnen Clubs gesprochen wurden, kann ich mich ohne jede Einschränkung anschließen. Der Panorama-Club aus Duisburg, der das Treffen im nächsten Jahr durchführen will, wird es sehr schwer haben, das Niveau in Bremen zu erreichen oder gar noch zu übertreffen. Was aber die Duisburger nicht daran hindern sollte, es trotzdem zu versuchen. *Stelter/Kaiser*

DSR-Programmierung-Teil II

Diesmal zeige ich Ihnen die Arbeitsweise der Routinen SAVE, OLD und DELETE. Ich gehe dabei nicht einfach streng durch das GROM von vorne nach hinten durch, sondern zeige die Routinen in ihrer von der Funktion her logischen Reihenfolge.
Zuerst müssen Sie wissen, wie die drei RAM-Segmente aussehen, wenn Sie ein Basic-Programm enthalten. Sie haben jeweils einen 8 Byte langen Header, dann folgt das Programm:

Bytes 0 & 1	5AA5	RAM-Segment enthält ein Basic-File
Byte 2	FF	Im RAM-Segment liegt ein Basic-Programm
Byte 3	..	nicht benutzt
Bytes 4 & 5	....	Programmlänge
Bytes 6 & 7	....	nicht benutzt
Bytes 8 bis Segmentende:		Basic-Programm

Wenn Sie das nun wissen, ist der Rest einfach. Zunächst die SAVE-Routine, die als erstes prüft, ob das Programm überhaupt in den RAM-Bereich hineingeht:

G6395	DCH	§>8350,§>835C	Vergleich Prg.-Länge mit Speicherl.
G6398	BS	GROM§>61C0	Zu lang: ERROR 4 (Buffer full)

Anschließend wird geprüft, ob bereits Basic-Files im Segment liegen. Wenn ja, werden sie einfach überschrieben. Wenn nein, muß neu initialisiert werden:

G639A	DCEQ	§>0000(§>835A),>5AA5	Basic-Files im RAM?
G63A1	BS	GROM§>63A6	Ja: weiter
G63A3	CALL	GROM§>646B	Nein: Routine für Init

Nun kann gesaved werden:

G63A6	DST	§>834A,§>835A	Segmentstartadr. kopieren
G63A9	DADD	§>834A,>0008	8 Bytes weiter startet das Progr.
G63AD	ST	§>0002(§>835A),>FF	Identifier für Programm
G63B3	DST	§>0004(§>835A),§>8350	Längenangabe i.d. Header
G63B9	MOVE	§>8350 TO §>0000(§>834A) FROM VDP*>834C	Programm ins Segment kopieren
G63C1	BR	GROM§>61DE	GSR-Return to Basic

Damit ist ein Basic-Programm gesaved. Die OLD-Routine, die die Programme wieder lädt, ist demnach ebenfalls sehr einfach:

G6365	DCEQ	§>0000(§>835A),>5AA5	Basic-Files im RAM?
G636C	BR	GROM§>61D0	Nein: ERROR 7 (File Error)
G636E	CEQ	§>0002(§>835A),>FF	Ist ein Programm im RAM?
G6374	BR	GROM§>61D0	Nein: ERROR 7 (File Error)
G6376	DST	§>835A,§>835A	Bereichsstartadr. kopieren
G6379	DADD	§>834A,§>0008	8 Bytes weiter startet das Programm
G637D	DCH	§>0004(§>835A),§>8350	Ist im VDP-RAM genug Platz f.d. Prg.?
G6383	BS	GROM§>61C0	Nein: ERROR 4 (Buffer full)
G6385	DST	§>834E,§>0004(§>835A)	Längenangabe kopieren
G638B	MOVE	§>834E TO VDP*>834C FROM §>0000(§>834A)	Programm ins VDP-RAM kopieren
G6393	BR	GROM§>61DE	GSR-Return to Bas.

Damit können Sie ein gesavtes Programm wieder zurückholen. Ist doch alles spielend einfach, oder? Deshalb jetzt noch die Routine DELETE. Sie fragt als erstes ab, ob der zu löschende Speicherbereich überhaupt ein Basic-File enthält:

G6433	DCEQ	§>0000(§>835A),>5AA5	Identifier Basic-Files
G643A	BR	GROM§>6469	Nein: GSR-Return to Basic
G643C	DCLR	§>0000 (§>835A)	Identifier löschen

Durch das Löschen des Identifiers wird der Bereich als leer ausgewiesen. Ist im Minimem kein Basic-File enthalten, so werden dort noch die entsprechenden Pointer auf die erste und letzte freie Speicherstelle des Segments geschrieben:

G6441	DCEQ	§>7000, >A55A	Assemblerfiles im 4K-Segment?
G6447	BR	GROM§>6469	Nein: GSR-Return to Basic
G6449	DCEQ	§>835A, >2000	War das File in der Low-Mem?
G644D	BR	GROM§>645D	Nein: Files waren in der Hi-Mem
G644F	DST	§>7022, >2000	Bereichsstartadr. für Assembler
G6455	DST	§>7024, >3FFF	Bereichsende für Assembler
G645B	BR	GROM§>6469	GSR-Return to Basic
G645D	DST	§>7026, >A000	Bereichsstartadr. für Assembler
G6463	DST	§>7028, >FFE0	Bereichsende für Assembler
G6469	BR	GROM§>61DE	GSR-Return to Basic

Nun fehlt Ihnen eigentlich nur noch die Routine zur Initialisierung des gewünschten Segments, die bei der SAVE-Funktion verwendet wurde.
Sie funktioniert folgendermaßen:
Soll die Speichererweiterung beschrieben werden und liegen in der Minimem Assemblerfiles, so wird das gewünschte Segment neu initialisiert und im Minimem die entsprechenden Pointer gesetzt. In allen anderen Fällen wird nur initialisiert.

G646B	DCEQ	§>835A, >7000	Wird die Speichererweiterung beschrieben?
G646F	BS	GROM§>6491	Nein: initialisieren
G6471	DCEQ	§>7000, >A55A	Assemblerfiles in der Minimem?
G6477	BR	GROM§>6491	Nein: initialisieren

»

```
G6479   DST    §>834A,        Pointeradresse
               >7022          Low-Mem
G647D   DCEQ   §>835A,        Low-Mem ange-
               >2000          sprochen
G6481   BR     GROM§>6487     Nein: weiter
G6483   DADD   §>835A,        Ja: Start bei >2004
               >0004
G6487   DCLR   §>0000         Pointer für Startadr.
               (§>834A)       löschen
G648C   DCLR   §>0000         Pointer für Endadr.
               (§>834A)       löschen
G6491   DST    §>0000(§       Identifier Basic-Files
               >835A),>5AA5
G6498   DCLR   §>0006         Current File Entry
               (§>835A)       Point
G649D   DCLR   §>0004         End-of-File-
               (§>835A)       Pointer
G64A2   CZ     §>834E         (maximale) Satzlän-
                              ge gleich 0?
G64A4   BR     GROM§>64AF     Nein: weiter
G64A6   ST     §>834E,>50     Ja: Default ist 80
G64A9   ST     VDP§>FFFA      Default auch in
               (§>8358),      den PAB
               §>834E
G64AF   ST     §>0003(§       Logical Record
               >835A),        Length
               §>834E
G64B5   ST     §>834A,        Statusbyte des
               §>834B         PAB kopieren
G64B8   AND    §>834A,>18     Bits FIX-VAR/DIS-
                              INT abspalten
G64BB   ST     §>0002         Als Filestatus
               (§>835A),      schreiben
               §>834A
G64C1   RTN
```

Wenn Sie übrigens genau hinsehen, merken Sie, daß in dieser Routine ein Programmierfehler von Texas Instruments vorliegt. Der Befehl auf G6483 sollte nämlich wohl mal DADD §>834A,>0004 heißen, sonst gibt das Ganze keinen rechten Sinn. Da ich im Minimem bereits schon einige andere Fehler gefunden habe, ist das auch durchaus möglich.

Sven Dyroff

BASICPROGRAMME EDITIEREN

Gibt es eine Möglichkeit, mit dem TI-Writer Basic-Programme zu editieren? Wenn nicht, gibt es Ihnen bekannte Software, die einen passenden Zeichensatz generieren kann und dessen Anwendung einfach ist, um z.B. Basic-Programme editieren zu können?
Dr. Volker Fuchs
7800 Freiburg

Mit dem TI-Writer lassen sich Basic-Programme schreiben.
Ihr Problem liegt darin, daß der TI-Writer für seine Files das DIS/VAR 80-Format benötigt, bei einem Basic-Programm fehlt jedoch diese Voraussetzung.
Um nun ein Basic-Programm in das DIS/VAR 80-Format umzusetzen, brauchen Sie nur zu schreiben: LIST "DSK1.FILE"
Das Basic-Programm FILE wird dann auf Laufwerk 1 in dem DIS/VAR 80-Format abgelegt und kann mit dem TI-Writer bearbeitet werden.
Um das Programm starten zu können, muß man es wieder in das Program Format zurücktransformieren.
Dazu benötigt man ein Programm mit dem Namen Convert.
Dieses Programm war vor etwa 2 Jahren noch als Modul erhältlich, dann wurde es aber nicht mehr im Handel angeboten.

DATEI VERSCHWUNDEN

Beim Testen eines selbstgeschriebenen Assembler Programms ist mir leider ein Mißgeschick mit meiner Diskette geschehen. Da das Programm eine PAB enthielt, wurde eine Datei mit wichtigen Texten gelöscht. Nach dem Überprüfen der Sektoren sah ich jedoch, daß der gesamte Text noch auf der Diskette enthalten ist. Lediglich die Datei existiert nicht mehr. Gibt es nun eine Möglichkeit, die Texte wieder in eine Datei zu holen, oder sind sie rettungslos verloren?
Steffen Allgaier
Obersdorf

Wie aus Ihrem Brief hervorgeht, sind Sie im Besitz eines Diskettenmonitors. Mit dessen Hilfe ist es möglich, die Texte wieder zu retten. Der Weg dazu ist jedoch mit Schweiß belegt. Eröffnen Sie zunächst eine Datei, die das gleiche Format, aber einen anderen Namen hat wie die ehemaligen Texte. Anschließend schließen Sie diese Datei wieder, ohne mit ihr etwas gearbeitet zu haben. Nun gehen Sie in den Diskettenmonitor.
Suchen Sie die Sektoren heraus, in denen die Texte stehen. Ändern Sie nun Sektor Null so ab, daß diese Sektoren als belegt gelten. Nun gehen Sie in den Sektor, der den Namen der Datei enthält, die Sie vorher eröffnet haben. Ändern Sie diesen Sektor so ab, daß nun die betreffenden Sektoren zu dieser gehören.
Haben Sie dabei keinen Fehler gemacht, so müßten nun die Texte in der neuen Datei enthalten sein und damit abrufbar.

HARD-DISK AN DEN TI

Ich betreibe mit meinem TI sehr viel Datenverwaltung. Seit einiger Zeit beginnen die Dateien einen solchen Umfang anzunehmen, daß selbst eine doppelseitige, doppelte Dichte-Diskette nicht mehr ausreicht. Auch die Geschwindigkeit der Laufwerke läßt bei diesem Umfang zu wünschen übrig. Nun habe ich folgende Frage an Sie: Ist es möglich, eine Hard-Disk an den TI anzuschließen? Was kostet eine solche in der Regel?
Egon Kukulenz
München

Selbstverständlich ist es möglich, eine Hard-Disk an den TI anzuschließen. Die Kosten schwanken hierbei zwischen zwei- und dreitausend Mark. Um einen genauen Preis zu erfahren, der auch sehr stark vom Dollar-Kurs abhängt, fragen Sie am besten bei den Händlern nach, die in der TI-REVUE inserieren.
So bietet z.B. die Firma Myrac, die auch den TI-Nachfolger Geneve vertreibt, eine Hard-Disk für den TI-99/4A an.

TASTATUR PRELLT, WAS TUN?

Seit langem verwende ich meinen TI auch als Textverarbeitungsmaschine. Noch nie gab es Probleme mit der Tastatur. Doch seit kürzerer Zeit beginnen einige Tasten zu prellen, d.h., sie sprechen auf einmaligen Druck mehrmals an. Ist meine Tastatur kaputt, da ich sie sehr beansprucht habe? Ist das ein normaler Verschleiß? Kann man sie reparieren, oder muß ich eine neue kaufen und die alte austauschen?
Egon Czestowa
Frankfurt

Durch Schreiben allein ist bisher noch keine Tastatur kaputt gegangen. Es handelt sich hier um eine ganz normale Verschmutzung. Selbst wenn Sie regelmäßig Staub wischen, bleibt immer ein wenig an den Stellen hängen, die Sie nicht erreichen. Eine solche Stelle sind die Kontakte der Tastatur. Sind diese verschmutzt, so kann es durchaus zum Prellen der Tastatur kommen. Um sie wieder zu reinigen, gehen Sie folgendermaßen vor: Entfernen Sie alle Tastenkappen. Dies geschieht durch einzelnes, senkrechtes Abziehen der Tasten. Reinigen Sie dann die freigelegten Kontakte mit einem Wattestäbchen. Nach dem Reinigen können Sie eventuell die Kontakte mit einem ölfreien Kontaktspray, z.B. Kontakt WL, einsprühen. Damit müßte Ihr Problem behoben sein. Sollte die Tastatur geschlossen sein, d.h., um jeden Kontakt eine Plastikpatrone, so ist das Reinigen sehr mühsam und oft aussichtslos. In diesem Fall kaufen Sie sich am besten eine neue Tastatur. Diese ist um rund 20,– DM erhältlich.

BETRIFFT: 2. EXTERNES LAUFWERK

Vor kurzem bin ich auf eine komplett eingerichtete Peri-Box gestoßen. Da ich bereits eine mit Laufwerk, 32K und RS 232 besitze, möchte ich nun das andere interne Laufwerk als 2. externes benutzen (DSK.2). Frage: Ist das überhaupt möglich, und wenn ja, was muß ich tun? Erreicht man mit 2 32K-Karten eine doppelte Kapazität?

Poloni Daniel
8254 Basadingen

Prinzipiell ist es möglich, bis zu drei Laufwerke über einen TI Controller anzusteuern. Sie können Ihr Laufwerk in der zweiten PBox lassen, lediglich die Codierung des zweiten Laufwerks muß angepaßt sein, damit das Laufwerk auch als DSK2. angesprochen werden kann.
Die Umcodierung muß man direkt am Laufwerk vornehmen. Bei den Shugart Laufwerken, die von TI angeboten wurden, befindet sich auf der Platinenoberseite, rechts oben, gleich neben der Anschlußstelle für das Flachbandkabel, eine Codierleiste.
Diese Codierleiste steckt in einem IC-Sockel und besteht aus 6 Metallstreifen, von denen 4 unterbrochen sind.
Auf der rechten Seite des IC-Sockels steht "HL 1 2 3". Die Zahlen 1, 2 und 3 stehen für die Laufwerksnummer, mit der das Laufwerk angesteuert werden soll.
Jetzt können Sie die Verbindung bei 1 lösen und die bei 2 wieder herstellen und schon ist das Laufwerk als DSK2 definiert.
Bei Ihrem Controller brauchen Sie dann nur noch beim 2. Ausgang das Flachbandkabel anzuschließen und mit dem 2. Laufwerk zu verbinden.
Mit einer zweiten 32K-Karte läßt sich keine doppelte Kapazität erreichen, da der TMS 9900 nur einen begrenzten Speicherplatz verwalten kann.
Die zweite Karte kann nur abwechselnd mit der ersten betrieben werden.
Jedoch sind dazu fundierte Software- und Hardwarekenntnisse nötig.

DRUCKER OHNE P-BOX

Ich besitze einen TI-99/4A mit Extended Basic Modul und einer 32K-Erweiterung. Diese habe ich als Bausatz in die Console eingebaut. Ich besitze jedoch keine P-Box, so daß sich mir die Frage aufdrängt, wie ich einen Drucker ohne großen finanziellen Aufwand an meinen Rechner anschließen kann. Nun meine Frage an Sie: Können Sie mir sagen, ob eine solche Schnittstelle existiert, die man direkt an den Computer anschließen kann, also ohne Box?

Andreas Bowe
Braunschweig

Selbstverständlich existieren einige Schnittstellen, die direkt an den Rechner angeschlossen werden können. Anfragen über Preise können Sie an die Händler richten, die in der TI-REVUE inserieren. Sollten Sie jedoch aber auch zu den Computerfreunden gehören, die den Lötkolben dem Bestellschein vorziehen, in diesem Fall die TI-REVUE 12/86 lesen. In dieser veröffentlichten wir eine Centronics Schnittstelle im Eigenbau.

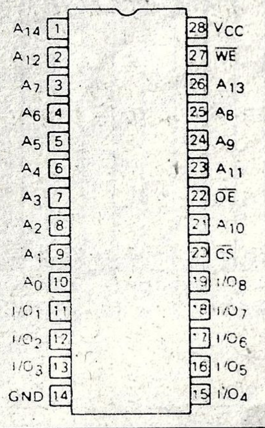
32K-ERWEITERUNG FÜR CA. 25 DM

Heute möchte ich Ihnen noch eine 32K-Byte-Speichererweiterung für den TI99/4A vorstellen. Der Vorteil dieser Schaltung ist ihre Unkompliziertheit und vor allem der extrem niedrige Preis.

262 144 BIT STATIC CMOS RAM

The µPD43256C/GU is a high speed, low power, 32768 words by 8 bits static CMOS RAM fabricated with advanced silicon-gate CMOS technology. The µPD43256C/GU is a low standby power device using n-channel memory cell with polysilicon resistors. Furthermore, a novel circuitry technique makes the µPD43256C/GU a high speed and low operating power device which requires no clock or refreshing to operate.
Minimum standby power is drawn by this device when CS is at a high level, independently of the other inputs level.
Data retention is guaranteed at a power supply voltage as low as 2 volts. (µPD43256C/GU-10L/12L/15L)
The µPD43256C is packaged in a standard 28-pin dual-in-line plastic package. The µPD43256GU is packaged in a mini flat package providing high density application.

- Single +5V Supply
- Fully Static Operation: No Clock or Refreshing required
- TTL Compatible: All Inputs and Outputs
- Commom I/O Using Three-State Output
- One Chip Select and One Output Enable Inputs for easy Application
- Fast Access Time:
  - µPD43256C/GU-10L 100ns MAX
  - µPD43256C/GU-12L 120ns MAX
  - µPD43256C/GU-15L 150ns MAX
- Low Power Dissipation:
  - Active 70mA MAX.
  - Standby µPD43256C/GU-10L/12L/15L 100µA MAX.
  - Data Retention 1µA MAX.
- Data Retention Voltage: 2V MIN. (µPD43256C/GU-10L/12L/15L)
- Standard 28-pin Plastic Package (µPD43256C)
- Miniflat package (µPD43256GU)



PIN NAMES

A_0-A_{14}	Address Input
I/O_1-I/O_8	Data Input/Output
$\overline{CS}$	Chip Select
$\overline{OE}$	Output Enable
$\overline{WE}$	Write Enable
V_{CC}	Power (+5 V)
GND	GND

Unkompliziert ist sie deshalb, weil bei einem Einbau in die Konsole lediglich 2 ICs benötigt werden. Falls Sie die Erweiterung nicht einbauen, sondern am I/O-Port Ihres Rechners anschließen wollen, kommen Sie mit insgesamt 3 ICs aus. Dazu aber später.

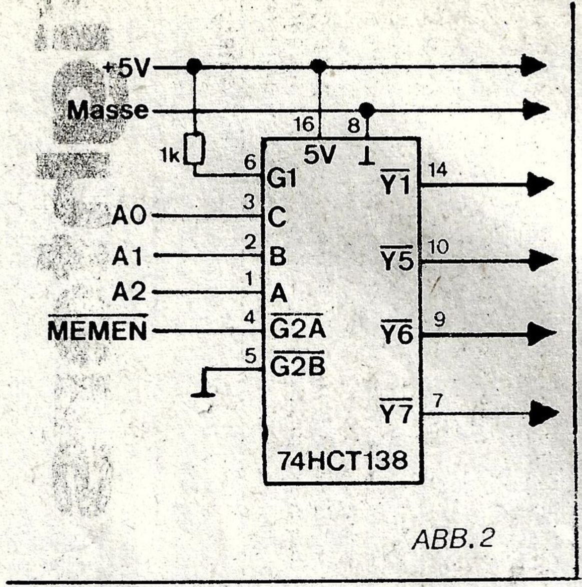


ABB.2

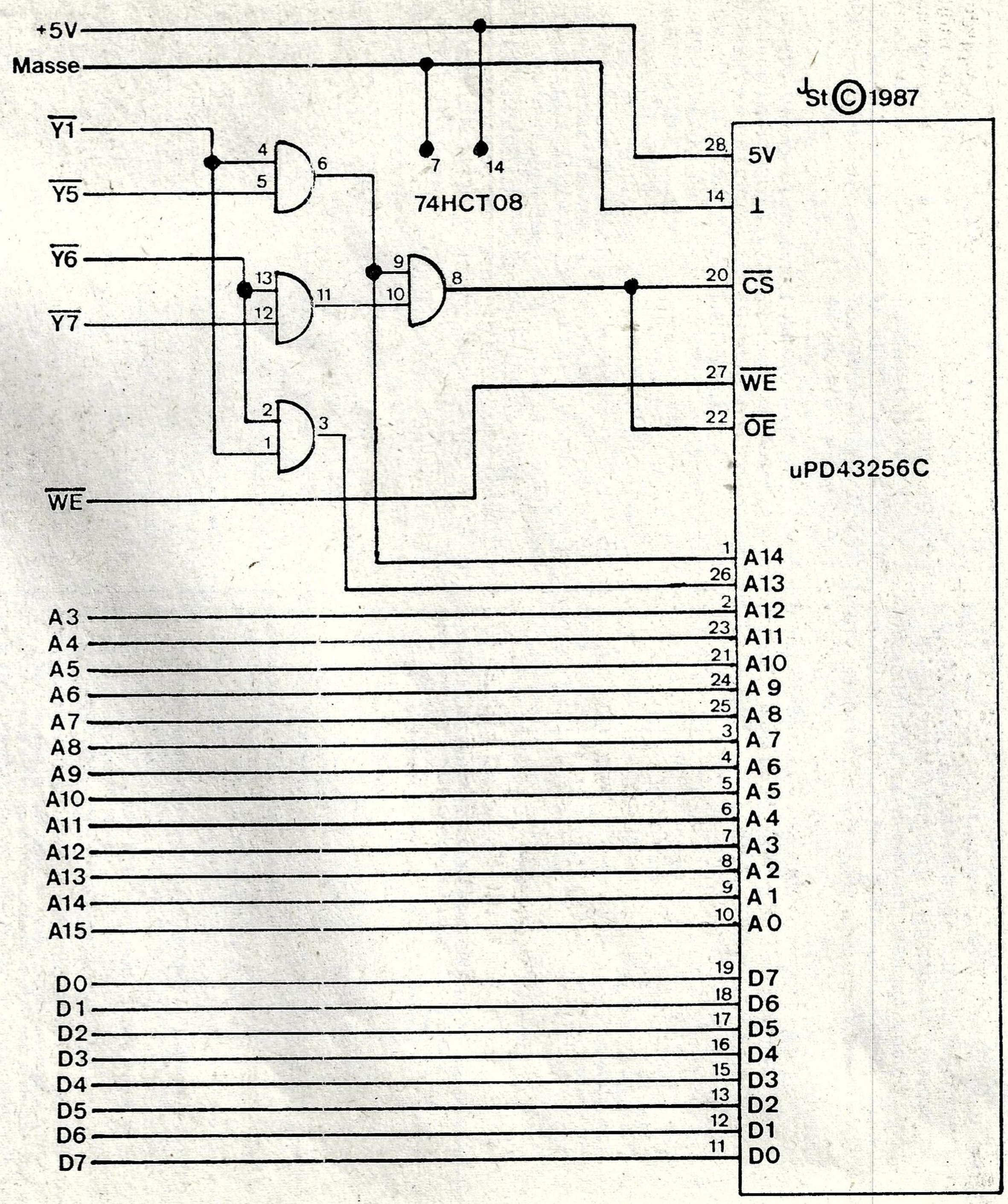


Tabelle A

	Platine (IC/Pin)	I/O-Port (Pin)		Platine (IC/Pin)	I/O-Port (Pin)
A0	-	31	D0	U614/11	37
A1	-	30	D1	/12	40
A2	-	20	D2	/13	39
A3	U503/ 7	10	D3	/14	42
A4	/ 5	7	D4	/15	35
A5	/ 3	5	D5	/16	38
A6	U509/ 7	29	D6	/17	36
A7	/ 5	17	D7	/18	34
A8	/ 9	14			
A9	/11	18	$\overline{Y1}$	U504/14	-
A10	U503/13	6	$\overline{Y2}$	/10	-
A11	/11	8	$\overline{Y3}$	/ 9	-
A12	/ 9	11	$\overline{Y4}$	/ 7	-
A13	U509/ 3	15			
A14	/13	16	$\overline{WE}$	U510/ 9	26
A15	U510/13	19	$\overline{MEMEN}$	-	32
+5Volt	U510/20	1			
Masse	/10	21,23,25,27			

(- = nicht benötigt)

Die Schaltung besteht in der Hauptsache aus einem statischen 32K*8 CMOS-RAM (NEC uPD43256C oder Hitachi HM62256 LP).
Diese Bausteine sind in der letzten Zeit im Preis so weit gefallen, daß sie zumindest nicht mehr teurer sind als 4 ICs der 8K*8 Typen, wie sie sowohl in den beiden Versionen der 16-Bit-Erweiterung (siehe TI-REVUE 7/87 und 8/87) als auch in der 8-Bit-Erweiterung (TI-REVUE 9/85) verwendet werden. Durch die Benutzung eines einzigen Speicherbausteines läßt sich diese Schaltung ohne Probleme auf einer ganz normalen Lochrasterplatine aufbauen, eine geätzte Platine erübrigt sich. Weiterhin ist lediglich ein 74HCT08 (evtl. 74LS08) erforderlich, um die 4 benötigten Bereiche >2000, >A000 >C000 und >E000 im Speicherbaustein zu selektieren. Dieses TTL-Gatter erzeugt neben den Adressen A13 und A14 auch das Chipselect-Signal (CS). Der Output-Enable Eingang (OE) des Speicherchips kann mit der CS-Leitung verbunden werden, das DBIN-Signal wird bei dieser Schaltung nicht benötigt.

DER EINBAU IST KEIN PROBLEM

Der Einbau in den TI ist ebenfalls für Leute mit etwas Lötefahrung kein Problem. Die Datenleitungen können an einem der folgenden ICs angeschlossen werden: U614 (74LS245), U615 (74LS373) oder U616 (74LS244). An jedem dieser ICs sind alle 8 Datenleitungen vorhanden (siehe Tabelle A).
Bitte beachten Sie: Bei fast allen Rechnern und Speichern ist D0 das LSB, nur der TI macht hier eine Ausnahme. Auch bei den Adressen ist normalerweise A0 die niederwertigste Adresse, beim TI ist das ebenfalls umgekehrt. Aus diesem Grunde stimmen auf dem Schaltplan die Adress- und Datenleitungen des

ORDNUNG IST DAS HALBE LEBEN

TI99 nicht mit denen des RAMs überein. Für die einwandfreie Funktion der Erweiterung ist dies jedoch ohne Bedeutung. Selbst wenn Sie alle Datenleitungen beliebig vertauschen, würden Sie dies beim Arbeiten des Rechners gar nicht bemerken. Dasselbe gilt auch für die Adressleitungen. Der Ordnung halber sollte man das aber beachten. Auch ist es empfehlenswert, zwischen +5 Volt und Masse an den beiden ICs je einen keramischen 100nF Kondensator einzulöten.
Die beschriebene Schaltung entspricht in ihrer Funktion und Geschwindigkeit der Original 32K-Karte von Texas Instruments. Allerdings ist der Aufwand an Bauteilen und somit der Preis auf ein kaum zu unterbietendes Minimum reduziert. Auch der Stromverbrauch der Schaltung belastet den TI so gut wie gar nicht, besonders dann, wenn Sie wie ich für das Gatter ein HCT-Typ benutzen, der heute preislich fast auf dem Niveau der LS-Typen liegt.
Wie anfangs bereits erwähnt, kann mit einem zusätzlichen IC die Schaltung auch am I/O-Port des TI99/4A betrieben werden. Dieser Decoder (74HCT138 bzw. 74LS 138) erzeugt aus A0, A1, A2 und MEMEN die Signale Y1, Y2, Y3 und Y4 (siehe Abb. 2), die im Rechner zwar vorhanden sind, aber nicht herausgeführt werden. Dieser Baustein ist ein Doppel des ICs U504 in der Konsole.
Die notwendigen Anschlüsse für beide Versionen können Sie der Tabelle A entnehmen. Irgendwelche Änderungen im Rechner sind nicht erforderlich. Bitte beachten Sie, daß sowohl der Speicher als auch die HCT-Gatter in CMOS-Technik aufgebaut sind. Sie können somit unter Umständen durch statische Aufladung zerstört werden.
Zum Schluß noch ein Wort zu den Preisen der ICs. Besonders bei dem benutzten 32K*8 CMOS-RAM gibt es sehr große Unterschiede. Sollten Sie Probleme haben, diesen Speicher oder auch die beiden HCT-Bausteine zu bekommen oder sollten diese bei Ihnen zu teuer sein, so kann Ihnen der TI-Workshop Wiesbaden selbstverständlich weiterhelfen. Da unser 512K-Byte RAM-Disk ebenfalls 16 dieser Speicher enthält, können wir, bedingt durch die Menge, diese RAMs preiswert einkaufen. Wir sind somit in der Lage, Ihnen die 2 bzw. 3 benötigten Bausteine für etwa DM 25 zu besorgen. Sollten Sie Interesse haben, hier die Adresse:
TI-Workshop Wiesbaden,
Eleonorenstraße 6,
6200 Wiesbaden
Telefonisch können Sie mich unter 06257/83247 erreichen, hier erfahren Sie dann auch die genauen Preise für die ICs, die sich ja öfters ändern (zum Glück meistens nach unten). *Jürgen Stelter*

Erfahrungsbericht zu Turbo-Pasc'99

Als Ergänzung zu einem Bericht in der TI-REVUE möchte ich meine Erfahrungen mit dem Turbo-Pasc'99 System und mit der Firma Softpoint an alle interessierten Leser weitergeben.

Kurz vor Weihnachten 1986 bestellte ich das Turbo-Pasc'99 System und erhielt es Ende Januar 1987. Mit Eifer begann ich sofort, ein Programm zu schreiben.
Nach der Erkenntnis, daß der enthaltene „Compiler" eigentlich gar kein Compiler ist, sondern aus dem Pascal Source Code einen Assembler Source Code erstellt, der dann mittels des normalen E/A weiterverarbeitet werden muß, lief anfänglich alles bestens.
Ich möchte hier gleich den Tip weitergeben, längere Programme in viele einzelne Module zu zerlegen und diese einzeln auszutesten.
Wenn man dies wie ich anfangs nicht macht, wird die Beseitigung von Programmierfehlern (nicht von Syntaxfehlern, das geht sehr schnell) sehr mühsam, da nach jeder Änderung der Pascal Code erst compiliert und dann assembliert werden muß. Dieser Vorgang dauert sehr lange und benötigt viel Speicherplatz, besonders wenn man nur 90KByte pro Diskette Platz hat.
In meinem Fall umfaßt der Pascal Source Code 52 Sektoren auf der Diskette, der erzeugte Assembler Source Code 197 und der Assembler Object Code 190 Sektoren. Das vom Linker erzeugte komprimierte File besteht dem gegenüber nur noch aus 55 Sektoren und kann direkt über die 'Run Program File' Option gestartet werden.
Während der Testphase kam ich dann zu dem Punkt, an dem Daten von Diskette zu lesen und wieder auf Diskette zu schreiben waren. Bei jedem Versuch gab es eine Fehlermeldung. Nach Stunden des Suchens und Lesens im Handbuch stellte sich heraus, daß

bei der Eröffnung von Relativ-Dateien die bereits bestehende Datei dieses Namens erst einmal gelöscht wird. Für diejenigen, die Turbo-Pascal (von Borland) kennen: von den beiden Befehlen 'rewrite' und 'reset' wurde für den TI nur der 'rewrite' Befehl verwirklicht. In der Praxis heißt dies, daß es unmöglich ist, Dateien zu ändern, es sei denn, sie sind klein genug, um neben dem Programm im Speicher Platz zu finden. Da ein Computer meines Erachtens auch dazu benutzt werden sollte, größere Datenmengen zu verwalten, ist jede Software, die dies nicht ermöglicht, sinnlos.

KEIN DIRECTORY!

Während diverser Versuche, einen Ausweg zu finden, traten noch weitere Unzulänglichkeiten zu Tage: es ist z.B. nicht möglich, das Directory einer Diskette zu lesen. Ferner ist es nur schwer möglich, Datensätze auf dem Bildschirm zu editieren, da der 'read'-Befehl immer die Zeile, auf der der Cursor steht, löscht (in XBasic kann man wenigstens über den 'size'-Befehl wählen, ob eine Eingabezeile gelöscht werden soll oder nicht).
Inzwischen ziemlich frustriert, versuchte ich mit der Firma Softpoint telefonisch Kontakt aufzunehmen. Bei vielleicht 5 – 6 Versuchen erhielt ich von der Dame am Telefon immer die Auskunft, daß die verantwortlichen Herren im Augenblick nicht zu erreichen wären, man mich aber umgehend zurückrufen würde. Nichts geschah! Erst als ich am Telefon etwas böser wurde (ich bin ein geduldiger Mensch), kam einige Tage später ein Rückruf. Man erkannte mein Problem und versprach, sich darum zu kümmern. Gleichzeitig bat man aber auch um Geduld, da man wegen der CeBit viel Arbeit hätte. Nach Abschluß der Messe würde man sich aber bestimmt wieder melden. Der geneigte Leser wird es vielleicht schon ahnen, es geschah nichts. Also faßte ich meine Punkte noch einmal in einem Brief zusammen. Anfang Mai erhielt ich dann einen Anruf, in dem mir mitgeteilt wurde, daß man sich meines Problems angenommen habe und am Montag d. 18. Mai 1987 eine neue Version des Turbo-Pasc'99 an mich schicken werde. Welche Freude! Aber nur für kurze Zeit. Statt eines Up-dates erhielt ich lediglich einen Brief, in dem mir mitgeteilt wurde, daß Relativdateien in Standard-Pascal auch nicht vorgesehen seien und man diese für den TI nur der Vollständigkeit wegen aufgenommen habe (warum eigentlich, wenn man nichts mit anfangen kann?).

KEIN UP-DATE!

Bezüglich meines Problems mit dem 'read'-Befehl verwies man auf das Softwareangebot 'Window-Manager', ebenfalls von Softpoint!
Der Brief schloß mit der Bemerkung, daß die Firma Softpoint nicht als Autor auftrete, sondern nur den Vertrieb vornehme. Später heißt es allerdings, daß Anpassung nur vorgenommen werden, wenn es wirtschaftlich für 'WipoSoft (=Softpoint?) sei.
Da ich 5 Monate lang keine Lösung gefunden habe und von der Firma Softpoint diesbezüglich wohl nichts mehr erwarten kann, wende ich mich an alle Leser der TI-REVUE: Kennt jemand eine Möglichkeit, Relativdateien unter dem Turbo-Pasc'99-System zu ändern?

Uwe Schulze

Gram-Loader-Modul

Eine Super-Erweiterung des Workshop Rheinland

Mit dem GRAM-Modul hat der User die Möglichkeit, ein ausgelesenes Modul von Diskette in diese Erweiterung einzuladen. Zum Beispiel das Extended Basic, Multiplan oder auch den TI-Writer. Selbstverständlich sind das noch nicht alle Programme, die eingelesen werden können. Zur Zeit sind etwa weitere 90 Module ausgelesen und können von Diskette in dieses Modul eingelesen werden. Im Klartext: Mit dieser Erweiterung entfällt das ewige „Modul Rein-Raus", denn man benötigt nur dieses eine Modul und kann alle bereits ausgelesene Module in diese Erweiterung einlesen und dann so benutzen, als hätte man dieses, von Diskette eingelesene, Modul im Modulport.
Die Ladezeit für die einzelnen Module ist auch relativ kurz. So beträgt die Einlesezeit für das Editor/Assembler Modul 23 Sekunden. Die Module können von Diskette wie auch von Cassette eingelesen werden, jedoch ist bei Cassettenanwahl die Ladezeit recht lange, so daß diese Möglichkeit nicht das „Gelbe vom Ei" ist, aber trotzdem. Die Erbauer haben auch an die immer noch zahlreichen Cassetten-Benutzer gedacht, was auch honoriert werden sollte. Ein Manko ist jedoch, daß das Modul nicht ohne Speichererweiterung und Disk lauffähig ist, was an der Loadersoftware hängt.
Das GRAM-Modul wird in einem Metallgehäuse geliefert und benötigt die gesamte schwarze Fläche vor dem Modulport, hat also etwa die Größe eines 3fach-Expanders der Firma Navarone.
Auf dem Modul befindet sich noch ein Steckplatz für ein weiteres Modul, welches hier eingesteckt werden kann, d.h., wenn man von Diskette das Editor/Assembler Modul geladen hat, und das aufgesteckte Modul, z.B. das Extended Basic ist, so ist durch die Auswahl des Punktes 5 in der "Loader"-Software ein problemloser Wechsel zum Extended Basic möglich. Will man nun zum Editor

TASTENDRUCK GENÜGT

Assembler Modul zurückkehren, so muß die Software nicht etwa neu geladen werden, sondern ist durch einen Druck auf einen Taster, der sich auf dem GRAM-Modul befindet, direkt anwählbar.
Der Steckplatz kann jedoch auch zum Auslesen von Modulen verwendet werden, was man wie folgt erreicht:

1. Man steckt das Modul, welches man auslesen will, in den freien Modulsockel.
2. Nun wählt man aus dem "GRAM-Loader" den Punkt 7 aus, welcher in ein Untermenü der "Loader-Software" verzweigt.
3. In dem Untermenü wählt man nun die Option 3 und braucht anschließend nur noch das Laufwerk und den gewünschten Dateinamen einzugeben.

Nun liest das GRAM-Modul das „aufgesteckte" Modul automatisch aus und speichert es unter dem angegebenen Namen auf dem angewählten Laufwerk ab. Dieser Vorgang dauert ca. 2 – 5 Minuten, je nach Modul. Jeder User kann also mit dem GRAM-Modul ein noch nicht ausgelesenes Modul kurz auslesen und das gekaufte Modul an-

schließend „verschrotten", da es ja auf Disk vorhanden ist.
Die Module, welche von Diskette eingelesen werden, werden über einen Loader ausgewählt. Dieser Loader ist von der Auswahlliste aus ansprechbar. Im Loader angelangt, hat man die Möglichkeit, das gewünschte Modul von DSK1 und DSK2 einzulesen, da diese beiden Ladearten vorgegeben sind. Mit der Option sechs der Auswahlliste im Loader kann man auch von DSK3 oder DSK4 das Modul einlesen, muß hier allerdings die Einlesequelle selbst eingeben, während sie bei DSK1 und DSK2 bereits vorgegeben ist.
Außerdem bietet der Loader noch die Möglichkeit, den Speicher zu löschen, zum Titelbild zurückzukehren oder sich mal kurz die Kontaktadressen des Workshops anzusehen.

FAZIT

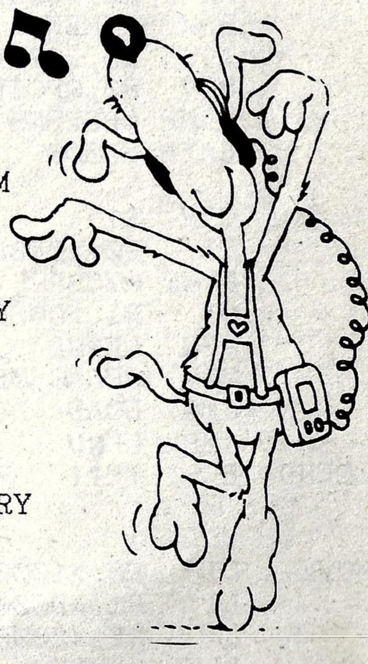
Das GRAM-Modul ist die sinnvollste Erweiterung für den Modulport des TIs. Bei einem Preis von 398,– DM ist die GRAM-Erweiterung sehr günstig beim TI-Workshop Rheinland oder bei uns, dem TI-Club Baunatal, zu erwerben. Bei einer Bestellung erhält der Käufer eine Garantie von sechs Monaten, so daß er kein Risiko beim Kauf eingeht. Das Modul sollte unserer Meinung nach jeder besitzen, da er damit seinen Modulport schont und somit das Leben seiner Konsole um einiges verlängert.

TI-Club Baunatal
Matthias Orf
und
Norbert Göbel

Programm-Lader für Gram-Modul in Maschinensprache

Besitzer des GRAM Moduls aus Heft 6/87 können den Basicprogramm Lader nun getrost zu den Akten legen. Das nachfolgende Programm lädt nach Angabe eines Filenamens die restlichen Files, falls vorhanden, automatisch. Das Erstellen eines DIS/VAR80 Files entfällt somit.
Bei Eingabe eines falschen Filenamens springt das Programm erneut in den Abfragemodus.
Die notwendigen Querverweise, um mittels des Programms SAVE des Editor/Assembler Pakets ein Programm File mit Namen UTIL1 zu erzeugen, wurden vorgesehen.

Reiner Ziegler



```
**********************************
*                                *
*  LADEPROGRAMM FUER FILES MIT   *
*  FOLGENDEM PROGRAMM-HEADER     *
*                                *
*  GROM_FILE: POSITION,LAENGE    *
*  ROM_FILE : POSITION,ØØØØ      *
*                                *
*  IM LADER MUSS NUR DIE LAENGE  *
*  DES NAMENS UND DAS ERSTE      *
*  FILE ANGEGEBEN WERDEN. BEI    *
*  MEHREREN FILES DUERFEN SICH   *
*  DIESE NUR IM LETZTEN BUCH-    *
*  STABEN UNTERSCHEIDEN Z.B.     *
*                                *
*  DSK1.TEST         DSK3.PASCAL *
*  DSK1.TESU         DSK3.PASCAM *
*  DSK1.TESV         DSK3.PASCAN *
*  DSK1.TESW         DSK3.PASCAO *
*  DSK1.TESX         DSK3.PASCAP *
*                                *
*    C JULI 87 BY R.ZIEGLER      *
*                                *
**********************************

        DEF LOAD,SLOAD          STARTNAMEN FUER PROGRAMM
        DEF SFIRST,SLAST        UND SAVE OPTION E/A

SLOAD
SFIRST B    @LOAD               MARKEN FUER SAVE UTILITY

        REF DSRLNK,VMBW         EXTERNE UNTERPROGRAMME
        REF VMBR,VSBR
        REF VSBW

PAB     EQU  >ØF8Ø              ADRESSE PAB IM VDP MEMORY
POINT   EQU  >8356              POINTER AUF NAME IM VDP

PABDAT DATA >ØØØØ               PAB DATENBLOCK FUER
       DATA >ØØØØ               FILE LESEN
       DATA >Ø5ØØ
       DATA >1ØØØ               VDP ADRESSE DER DATEN
       DATA >ØØØØ
```

```
START  DATA >2ØØ8           LAENGE DATEI
       DATA >ØØØØ
TEXT   TEXT 'DSK'           PLATZ FUER LAENGE NAME
TEXTNA TEXT 'XXXXXXXXXXXX'  UND FILE NAME
MYREG  BSS  32              BUFFER FUER EIGENEN REGISTERSATZ
GADR   DATA >6892           GROM RUECKSPRUNGADRESSE
FLAG   DATA >ØØØØ           FLAG
       EVEN

LOAD   LWPI MYREG           EIGENEN REGISTERSATZ LADEN
       BL   @INPUT          FILE NAMEN EINLESEN MAX 1Ø ZEICHEN
LLOOP  BL   @LESEN          DATEI LESEN MAX 34 SEKTOREN
       SETO @FLAG           FLAG SETZEN
       BL   @MOVE           DATEN INS GRAM ODER RAM KOPIEREN
       LI   R5,TEXT-1       POINTER AUF LAENGE FILENAME
       MOV  *R5,R6          FILELAENGE LESEN
       A    R6,R5           POINTER AUF ENDE FILENAME POSITIONIEREN
       MOVB *R5,R6          ASCII ZEICHEN LESEN
       AI   R6,>Ø1ØØ        UM EINS INKREMENTIEREN
       MOVB R6,*R5          KORRIGIERTEN FILENNAMEN ZURUECKSCHREIBEN
       JMP  LLOOP           NAECHSTES FILE EINLESEN

LEND   CLR  @>837C          STATUS REGISTER LOESCHEN
       BLWP @>ØØØØ          SOFTWARE RESET


* UNTER-PROGRAMME

LESEN  LI   RØ,PAB          VDP ADRESSE DES PAB BLOCKS NACH RØ
       LI   R1,PABDAT       ADRESSE PAB DATENBLOCK NACH R1
       LI   R3,START+2      LAENGE FILENAMEN LESEN
       MOV  *R3,R2          UND NACH R2 GEBEN
       AI   R2,>ØØØE        RESTLAENGE PAB BLOCK ADDIEREN
       BLWP @VMBW           INS VDP RAM SCHREIBEN
       AI   RØ,>ØØØD        POINT AUF NAMENSLAENGE
       MOV  RØ,@POINT       IM VDP RAM SETZEN
       BLWP @DSRLNK         AUSFUEHRUNG DATEI LESEN
       DATA >ØØØ8

       CLR  R1              R1 LOESCHEN
       LI   RØ,>ØF85        POINTER AUF ERRORKODE IM VDP
       BLWP @VSBR           EIN BYTE LESEN
       CI   R1,>ØØØØ        FEHLER BEIM LESEN
       JEQ  LEEND           NEIN, ENDE LESEN
       MOV  @FLAG,@FLAG     UEBERHAUPT SCHON EINE DATEI GELESEN ?
       JEQ  LOAD            WENN NEIN NEUBEGIN
       JMP  LEND            SONST ALLE FILES GELADEN, PROGRAMMENDE
LEEND  B    *R11            RUECKSPRUNG INS HAUPTPROGRAMM

MOVE   LI   RØ,>1ØØØ        ADRESSE DATEN IM VDP MEMORY
       LI   R1,>DØØØ        BUFFER FUER DATEN
       LI   R2,>2ØØ6        GESAMTLAENGE DATEN
       BLWP @VMBR           BYTES AUS VDP RAM LESEN

       MOV  @>DØØ2,R2       DATEILAENGE NACH R2
       JNE  GLOAD           UNGLEICH Ø, ALSO GROM
```

```
         MOV   @>DØØØ,R2            RAM STARTADRESSE
         MOV   R2,R4                HEX 2ØØØ BYTES
         AI    R4,>2ØØØ             ADDIEREN
         LI    R3,>DØØ4             DATENBEGIN NACH R3
RLOOP    MOV   *R3+,*R2+            DATEN UMSCHAUFELN
         C     R2,R4                ALLE DATEN UMKOPIERT
         JNE   RLOOP                WENN NEIN WEITER BIS FERTIG
         JMP   MEND                 PROGRAMMENDE

GLOAD    MOV   @>DØØØ,R1            GRAM STARTADRESSE FUER DATEN
         LI    R3,>DØØ4             DATENBEGIN NACH R3
         MOV   @>DØØ2,R2            LAENGE DATEI NACH R2
         MOVB  R1,@>9CØ2            HIGH BYTE DER ADRESSE SENDEN
         SWPB  R1                   BYTES VERTAUSCHEN
         MOVB  R1,@>9CØ2            LOW BYTE DER ADRESSE SENDEN
         SWPB  R1                   ADRESSE WIEDER KORRIGIEREN
GLOOP    MOVB  *R3+,@>9CØØ          BYTE INS GRAM KOPIEREN
         DEC   R2                   ANZAHL MINUS EINS
         JNE   GLOOP                SENDEN BIS ALLE BYTE KOPIERT
         MOVB  @GADR,@>9CØ6         GROMADRESSE
         MOVB  @GADR+1,@>9CØ6       KORRIGIEREN
MEND     B     *R11                 UNTERPROGRAMM ENDE

         COPY "DSK1.LOAD1;S"        PROGRAMM TEIL 2

SLAST    END
```

```
TEXTH    TEXT '                                  '
         TEXT '                                  '
         TEXT '                                  '
         TEXT '          LADER FUER GRAM-        '
         TEXT '          UND ROM-FILES           '
         TEXT '                                  '
         TEXT '          HEADER:                 '
         TEXT '                                  '
         TEXT '          GROM: POSITION,LAENGE   '
         TEXT '          ROM : POSITION,ØØØØ     '
         TEXT '                                  '
         TEXT '          COPYRIGHT JULI 1987     '
         TEXT '          BY R.ZIEGLER MOOSBURG   '


BUFF1    BSS  2
BUFF2    BSS  2

TEXTM    TEXT '        FILE : DSK
ENTER    BYTE >ØD
DEL      BYTE >Ø8

INPUT    MOV  R11,@BUFF1            RUECKSPRUNG SICHERN
         BL   @TXTWR                TEXTAUSGABE COPYRIGHT
         DATA Ø,TEXTH,416
```

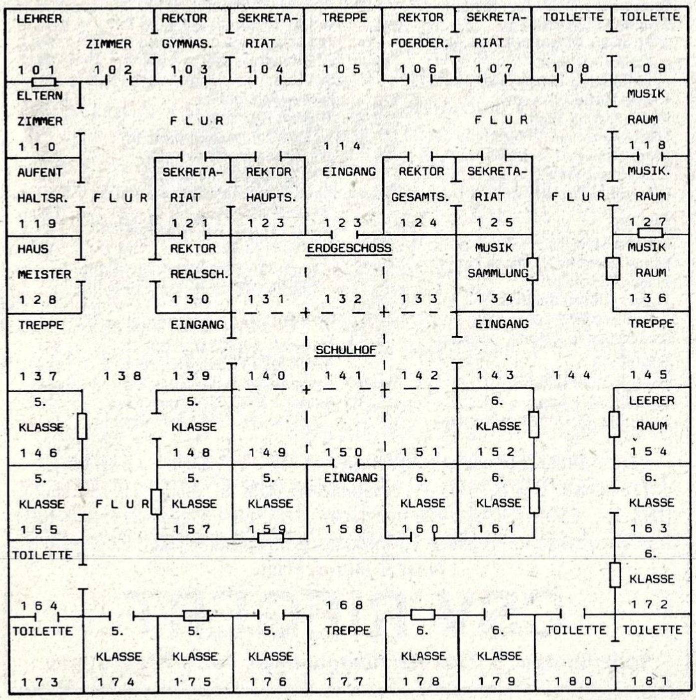
```
INPUT5 BL   @TXTWR          TEXTAUSGABE
       DATA 6Ø9,TEXTM,34    FILE : DSK
       LI   RØ,626          CURSOR POSITIONIEREN
       CLR  R5              ZEICHENZAEHLER LOESCHEN
       LI   R3,TEXTNA       POINTER AUF FILENAME
       LI   R1,>1EØØ        ZEICHEN FUER CURSOR LADEN
       BLWP @VSBW           UND AUSGEBEN
INPUT2 BL   @KEY            ABFRAGE TASTATUR
       CI   R5,Ø            SCHON EIN ZEICHEN GELESEN
       JEQ  INPUT4          WENN NEIN NACH INPUT4
       CB   @ENTER,R1       FALLS JA ENTER GEDRUECKT ?
       JEQ  INEND           WENN JA PROGRAMMENDE
       CB   @DEL,R1         DELETE GEDRUECKT ?
       JNE  INPUT4          WENN NEIN NACH INPUT4
       DEC  R5              BEI JA, ZEICHENZAEHLER MINUS EINS
       DEC  R3              POINTER AUF NAME DEKREMENTIEREN
       LI   R1,>2ØØØ        BLANK AUSGEBEN ZUM ZEICHENLOESCHEN
       BLWP @VSBW           AUSFUEHRUNG
       DEC  RØ              CURSOR NEU
       LI   R1,>1EØØ        POSITIONIEREN
       BLWP @VSBW           AUSFUEHRUNG
       JMP  INPUT2          SPRUNG ZUR EINLESEROUTINE
INPUT4 ANDI R1,>FFØØ        LOW BYTE LOESCHEN
       CI   R1,>21ØØ        ZEICHEN KLEINER ALS !
       JLE  INPUT2          SPRUNG ZUR EINGABE
       CI   R1,>5BØØ        ZEICHEN GROESSER ALS [
       JHE  INPUT2          SPRUNG ZUR EINGABE
       BLWP @VSBW           SONST DARSTELLEN DES ZEICHENS
INPUT1 INC  R5              ZEICHENZAEHLER PLUS EINS
       INC  RØ              BILDSCHIRMADRESSE ERHOEHEN
       MOVB R1,*R3+         ZEICHEN NACH FILENNAMENBUFFER KOPIEREN
       LI   R1,>1EØØ        CURSOR LADEN UND
       BLWP @VSBW           AUSGEBEN
       CI   R5,13           SCHON 12 ZEICHEN EINGELESEN ?
       JHE  INPUT5          FALLS 13 ODER MEHR NACH INPUT5
       JMP  INPUT2          SONST ZUR ROUTINE EINLESEN
INEND  AI   R5,3            GESAMTLAENGE KORRIGIEREN WEGEN "DSK"
       MOV  R5,@TEXT-1      WERT IN BUFFER NAMENSLAENGE KOPIEREN
       MOV  @BUFF1,R1Ø      RUECKSPRUNG HOLEN
       B    *R1Ø            ZURUECK INS HAUPTPROGRAMM

KEY    MOV  R11,@BUFF2      RUECKSPRUNG SICHERN
       LWPI >83EØ           WORKSPACE LADEN
LZ     CLR  @>8374          GANZE TASTATUR ABFRAGEN
       BL   @>ØØØE          TASTATURROUTINE IM ROM
       MOVB @>837C,R1       TASTE GEDRUECKT
       JEQ  LZ              FALLS NEIN, WARTEN
       LWPI MYREG           ALTEN WORKSPACE LADEN
       MOVB @>8375,R1       TASTENCODE NACH R1
       ANDI R1,>FFØØ        LINKES BYTE AUSBLENDEN
       MOV  @BUFF2,R1Ø      RUECKSPRUNG HOLEN
       B    *R1Ø            ZURUECK INS HAUPTPROGRAMM

TXTWR  MOV  *R11+,RØ        TEXTSTART NACH RØ
       MOV  *R11+,R1        POINTER AUF TEXT NACH R1
       MOV  *R11+,R2        LAENGE TEXT NACH R2
       BLWP @VMBW           AUSFUEHRUNG
       RT                   ZURUECK INS HAUPTPROGRAMM
```

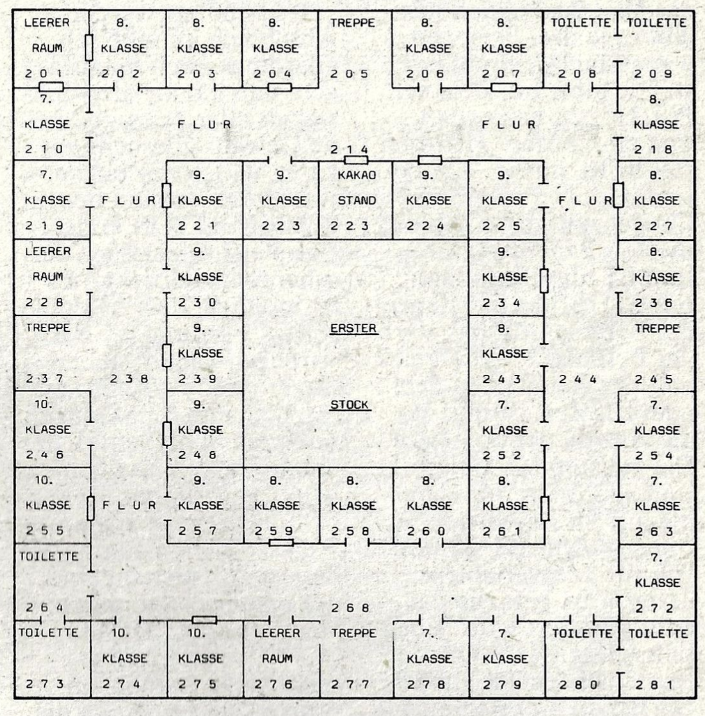
Die Lösung zum Schulhaus-Adventure

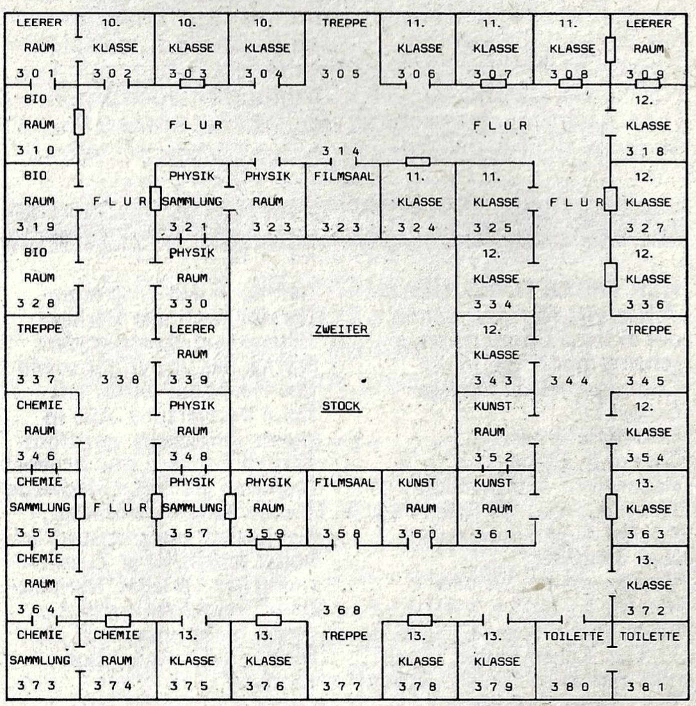
Nun ist es doch einige Zeit her, daß in der TI-REVUE Adventures abgedruckt wurden. Zu einem von diesen möchten wir nun die Lösung präsentieren. Wer sie nehmen will, Ok. Wer nicht, überliest am besten den Artikel. Noch etwas zum Bericht. Er ist nicht dazu gedacht, die Spannung zu nehmen, sondern vielmehr dazu, denen zu helfen, die in der Mitte stecken und nicht weiterkommen. Der Lösungsweg beschreibt das Adventure „Schulhaus".

Alle Gegenstände, die in dem Schulhaus verstreut liegen, haben eine Nummer, die den Raumnummern entspricht (s. Karte). Ebenso verhält es sich mit den Personen. Der Ort der Gegenstände ist in der Variablen G(X,Y) festgehalten,wobei X für die Art des Gegenstandes und Y für die Nummer des Gegenstandes steht. Die Gegenstände sind Zigaretten (X=1), Colabüchsen (X=2), Kuchenstücke (X=3), Pausenbrote (X=4), Dietriche (X=5) und der Schulranzen (X=6). Der Wert, den die Variable annimmt ist der Raum, in dem sie liegt. Ein Beispiel: G(%,2) =165 heißt, daß der zweite (Y=2) von drei Dietrichen (X=5) im Raum Nr. 165 liegt, in diesem Fall auf dem Flur. Mit einer Zeile "459 PRINT G (6,1)" erfährt man den Ort, an dem der Schulranzen liegt. Mit "PRINT G (5,1)" erfährt man, wo einer der Dietriche liegt. Die Variable für einen Gegenstand, den man bei sich hat, nimmt den Wert 400 (vierhundert) an. Mit "G(5,1)=400" kann man sich also von Anfang an einen Dietrich zuschanzen.

Sie können selbstverständlich auch versuchen, das Spiel ohne „Schummeln" durchzuspielen. Dabei gibt es folgende Dinge zu beachten: Es gibt zwei Arten von Personen, „gute" und „böse". Die „guten" Personen geben einem sofort Auskunft,







die „bösen" Personen verlangen etwas für ihre Antwort oder schicken sie auf den Hof. Es gibt jedoch auch bei den „guten" Personen Einschränkungen, denn Lehrer und Hausmeister schicken Sie auf den Schulhof, wenn Sie nach etwas anderem als dem Schulranzen fragen. Selbstverständlich können Sie sich auch vorher von der Position der Personen erkundigen. Die Variable für Personen ist P(X,Y), wobei X=1 für einen „bösen" Jungen steht, dem man Zigaretten oder die Colabüchse geben muß, bevor er antwortet. X=2 steht für den „bösen" Lehrer und X=3 für den „bösen" Hausmeister. X=4 steht für den „guten" Jungen, X=5 für den Lehrer, X=6 für den Hausmeister und X=7 für ➔

das Mädchen. Auch hier gibt es je drei Personen, wie es auch je drei Gegenstände gibt. Ein Beispiel: P(3,3)=228 heißt, daß sich ein „böser" Hausmeister in Raum Nr. 228 befindet.
Einige weitere Variablen, die von Bedeutung sind, sind E$, die Eingabevariable, D$, das Substantiv des Befehls, A$, das Verb des Befehls,O, die Nummer des Raumes, in dem man zur Zeit ist und M, die Anzahl der Spielzüge. Die meisten Variablen sind allerdings nur Variable mit lokaler Bedeutung, so sind L$, S$ und W$ nur Lesevariablen, deren Wert beinahe dauernd geändert wird. Ebenfalls wichtig sind O$, der Name des Raumes und G$(B), die Bezeichnung des Gegenstandes der vorhanden ist, P$, der Name der Person, die sich an Ihrem Ort befindet und N$(N(X)), die die möglichen Richtungen angeben, da diese alle für die Anzeige benötigt werden. Abschließend muß ich allerdings noch hinzufügen, daß ich für die Richtigkeit aller Aussagen keine Gewähr übernehmen kann, da ich nur eine etwas ältere Version des Spiels habe. (Die endgültige Version habe ich versehentlich gelöscht.) Es ist möglich, daß die endgültige Fassung, die Sie vorliegen haben, sieben Gegenstände enthält: zusätzlich zum Schulranzen noch ein Sportbeutel, der dann die Nummer 6 erhalten hat, wobei der Schulranzen die Nummer 7 hat. Welche Fassung Sie haben, läßt sich an einer der ersten Zeilen sehen. Heißt es dort "DIM G (6,3)", so gibt es nur sechs Gegenstände, ansonsten sieben. Auch die Bezeichnung der Räume und ob sie offen oder verschlossen sind, mag sich verändert haben, die Anordnung ist aber prinzipiell so wie auf der Karte.

Christoph Heuer

IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Geschäftsführer: Werner E. Seibt
Redaktion: Alfons Bauer, Thomas Dehn, Christoph Kögler, Jürgen Puschmann, Uwe Schröder
Verantwortlich für den Inhalt:
MSX-Teil: Werner E. Seibt
TI-Teil: Oliver Huber (Technik) Hartmut König (Listings)
Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d
Alle: Postfach 1161
8044 Lohhof.
Anfragen bitte nur schriftlich
Es gilt die Honorarliste des Verlages.

Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.

Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

©by CA Verlag, Heßstraße 90, 8000 München 40 SPS und Autoren

Die TI-REVUE soll weiterleben! Wir suchen noch gute Autoren!

TI99/4A

BRANDHEISSE KNÜLLERPREISE: (SOLANGE VORRAT REICHT)

Hardware

Platine TI-99/4A (z.Teil defekt)	2 St. 39,–
Tastatur TI-99/4A	39,–
RS 232–Karte (Atronic)	249,–
Disccontroller DSDD (Corcomp)	349,–
P-Code-Karte (gebraucht)	399,–
80-Zeichen-Karte (Mechatronic)	449,–
Diskettenlaufwerk DSDD mit Einbausatz für Peripheriebox	679,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle	249,–
Ext. 128 K-Erweiterung mit Centronicsschnittstelle + Druckerkabel + Brotherdrucker HR 5	449,–
dto. + Epsondrucker LX 800	849,–
dto. + Epsondrucker IX 800	1049,–
dto. + Epsondrucker LQ 800	1299,–
Axiom Centronicsinterface extern	129,–
RGB-Modulator mit Scartstecker	179,–
RGB-Modulator + Farbmonitor 1084	849,–
Grünmonitor Thomson (35 Mhz, Ton) mit Monitorkabel für TI-99/4 A	249,–
MBX-Sprachsteuereinheit + Baseball + Sewermania + Meteor Belt	229,–
Grafiktablett Supersketch + Buck Rogers + Defender	129,–
Supersketch + Defender + Dig Dug + Donkey Kong + Statistik	149,–
WIPO Mouse 99 anschlußfertig mit dt. Handbuch + Diskette	149,–
Akustikkoppler Dataphon S 21 d + Kabel für RS 232 Karte	279,–
Modulexpander 3-fach	125,–
Staubschutzhaube für TI-99/4A	20,–
Fernbedienung (2 Joysticks)	39,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter)	105,–

Bücher

Anwender-Handbuch Turbo Pascal dt.	39,–
Editor/Assembler Handbuch dt.	98,–
TI-99/4A Intern dt.	38,–
TI-Basic & Extended Basic	48,–
Assembler-Kurs (99er-Journal)	59,–
Introduction to Assembly Language	20,–

Software

Extended Basic II Plus	199,–
Mini Memory + Assemblerbuch dt.	199,–
Editor/Assembler (32 K + Disc notw.)	129,–
TI-Logo II (32 K + Disc notwendig)	149,–
Diskfixermodul (Navarone)	99,–
Database management (Navarone)	69,–
Terminal Emulator II + Statistik	59,–
Datenverwaltung/Analyse + Statistik	69,–
TI-99/4 A Flyer (Flugsimulation)	49,–

Modulgruppe A

Attack,Munchman, Othello, Invaders, Chisholm Trail, Tombstone City, Defender, Dig Dug — je 19,–

Modulgruppe B

Bigfoot, Balckjack & Poker, Espial, Fathom, Indoor Soccer, Moon Mine, Treasure Island, Donkey Kong, Jungle Hunt, Moon Patrol, Statistik — je 29,–

Modulgruppe C

Adventuremodul, Buck Rogers, Congo Bongo, Burgertime, Microsurgeon, Moonsweeper, Parsec, Return to Pirates Isle, Tunnels of doom, Video Chess, Protector II, Shamus — je 39,–

Je 1 Modul aus Gruppe A und B	44,–
Je 1 Modul aus Gruppe A und C	54,–
Je 1 Modul aus Gruppe B und C	63,–
Je 1 Modul aus Gruppe A, B und C	81,–
Popeye, Pole Position	je 49,–
TI-Artist (Diskette)	79,–
TI-Artist + Companiumdisketten 1,2 und 3 + Display Master	229,–
Graf-X + 3 Compan.disketten	199,–
Turbo Pascal (Diskette) + Anwenderhandbuch deutsch	149,–
Window Manager für Turbo Pascal	99,–
C 99 Compiler (Diskette)	99,–

Komplettübersicht über weiteres Zubehör + Modulpakete in unserer Preisliste !!!!!

Commodore / Schneider

Commodore C 128 D	969,–
Amiga 500 + Farbmonitor 1084	1.649,–
Amiga 2000 + Farbmonitor 1084	2.899,–
Schneider CPC 6128 Grünmonitor	729,–
CPC 6128 Farbmonitor	1.149,–

Versandkostenpauschale (Warenwert bis DM 1.000,– / darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–). Lieferung nur gegen NN oder Vorauskasse; Ausland nur Vorauskasse. Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstr. 5, 7324 Rechberghausen, Tel. (07161) 52889

Scrabble 86

Bei Scrabble86 handelt es sich um das fast gleichnamige Spiel. Bis zu 4 Spieler können hier mitspielen. Spieler 1 und 3 benutzen Joystick 1 und Spieler 2 und 4 Joystick 2. Die Nameneintragung zu Beginn des Spieles erfolgt im Kleinschreibemodus. Das Leerzeichen wird mit der offenen geschweiften Klammer simuliert. Einen Namen, der nur aus solchen „Leerzeichen" besteht, überspringt das Programm bei der Wortgenerierung. Der Spieler, der momentan dran ist, erhält einen abgesetzten Namenszug und der Cursor zeigt auf seine Buchstaben. Um einen Buchstaben aufzunehmen, muß die Spitze des Cursors über den betreffenden Buchstaben geführt und anschließend der Feuerknopf gedrückt werden. Der Buchstabe wird abgesetzt dargestellt und es kann nun mit dem Joystick zur betreffenden Stelle auf dem Spielbrett gefahren werden. Ein erneuter Druck auf den Feuerknopf plaziert den Buchstaben an der Cursorspitze. Jeder Spieler kann sechs Buchstaben auf dem Spielbrett plazieren. Nachdem ein Buchstabe plaziert wurde, wird an der nun freien Stelle unterhalb des Namens ein neuer Buchstabe dargestellt.

Um einen Buchstaben gegen einen zufällig generierten Buchstaben auszutauschen, muß zuerst 5 gedrückt werden, damit 2 neue Icons erscheinen. Mit dem Cursor wird nun das Icon One Swap angeklickt und der Cursor steht nun in der Reihe der Buchstaben des betreffenden Spielers. Mit dem Joystick kann ein Buchstabe ausgewählt werden und dieser wird dann durch einen neuen ersetzt, wenn der Feuerknopf gedrückt wird. Jeder Spieler kann bei einer Wortgenerierung sechsmal einen Buchstaben tauschen. Um einen schon plazierten Buchstaben wieder zu entfernen, muß das Icon One Back angelickt werden (evtl. vorher 5 drücken). Hierbei wird der Buchstabe, der zu letzt plaziert wurde, wieder entfernt und es wird auch die Punktzahl, die damit erreicht wurde, wieder abgezogen. Die Punktzahl wird um den Wert, der auf dem auserwählten Feld steht, mal 10 addiert. Ein Fragezeichen ergibt eine zufällige Punktzahl zwischen 0 und 100. Wenn kein Buchstabe mehr plaziert werden kann, so muß das Icon No Move angeklickt werden, damit der nächste Spieler weitermachen kann. Wenn ein Joker als Buchstaben auserwählt wurde und dieser dann auf dem Spielbrett plaziert wurde, dann kann man auswählen, welcher Buchstabe der Joker werden soll. Es wird einfach die entsprechende Taste auf der Tastatur gedrückt. Um das Spiel zu beenden, muß einfach das Icon End angeklickt werden und das Programm gibt die Plazierung der einzelnen Spieler mit deren Punktzahl aus. Die Leertaste beendet dann das Programm.

```
1 !     ****************
2 !
3 !   *** SCRABBLE86 ***
4 !
5 !  *** COPYRIGHT BY ***
6 !
7 ! *** JOERG LESSING  ***
8 !
9 !************************
1Ø CALL SCREEN(2):: CALL MAG
NIFY(3):: ON BREAK NEXT :: F
OR I=33 TO 143 :: READ I$ ::
 CALL CHAR(I,I$):: NEXT I ::
 FOR I=1 TO 24 :: READ I$ ::
 FOR I1=1 TO 9 :: CALL H(I,Ø
+I1,ASC(SEG$(I$,I1,1)))
11 DATA Ø1Ø1Ø1Ø1Ø1Ø1Ø1Ø1,FF,
Ø1,ØØØØØØØ3Ø1ØØØ1Ø3,Ø7ØF1FØE
Ø4,ØØØØØØF8F8F8F8F8,D888,Ø7Ø
7Ø7Ø7Ø7Ø7Ø7Ø7,EØEØEØEØEØEØEØ
E,ØØØØØØØØØØFFFFFF,FFFFFF
12 DATA ØØØØØØØØØØØØ7Ø7Ø7,ØØØ
ØØØØØØØEØEØE,Ø7Ø7Ø7,EØEØE,ØØ
3C242C34243C,ØØ18Ø8Ø8Ø83C,
ØØ3CØ43C2Ø2Ø3C,ØØ3CØ43CØ4Ø43
C,ØØ242424 3CØ4Ø4,ØØ3C2Ø3CØ4Ø
43C
13 DATA ØØ3C2Ø3C24243C,ØØ3CØ
4Ø4Ø4Ø4Ø4,ØØ3C243C24243C,ØØ3
C24243CØ43C,ØØ7E427E424242,Ø
Ø7C427C42427C,ØØ7E4Ø4Ø4Ø4Ø7E
,ØØ7C424242427C,ØØ7E4Ø7E4Ø4Ø
7E
14 DATA ØØ7E4Ø7E4Ø4Ø4,ØØ7E4Ø
4Ø4Ø467E,ØØ42427E424242,ØØ3C
1Ø1Ø1Ø1Ø3C,ØØ7EØ2Ø2Ø2427E,ØØ
4E5Ø6Ø6Ø5Ø4E,ØØ4Ø4Ø4Ø4Ø4Ø7E,
ØØ42665A424242,ØØ4262524A464
2
15 DATA ØØ7E424242427E,ØØ7C4
27C4Ø4Ø4,ØØ7E42424A467E,ØØ7C
427C484442,ØØ7E4Ø4Ø7EØ27E,ØØ
7E1Ø1Ø1Ø1Ø1,ØØ424242424 27E,Ø
Ø4444442828 1,ØØ44444454547C
16 DATA ØØ4224181824 42,ØØ444
428 1Ø1Ø1,ØØ7EØ4Ø81Ø2Ø7E,,A54
2BDØ4Ø4A55AA5,ØØØØØØØØØØØØØ1,
ØØ7E425E425E4,FFØ11129457D45
Ø1,FFØ179457945 79Ø1
17 DATA FFØ139454145 39Ø1,FFØ
17945454579Ø1,FFØ17D4179417D
Ø1,FFØ17D41794141Ø1,FFØ13D41
4D453DØ1,FFØ145457D4545Ø1,FF
Ø13911111139Ø1,FFØ17DØ5Ø5453
9Ø1
18 DATA FFØ14549714945Ø1,FFØ
141414141 7DØ1,FFØ1456D554545
Ø1,FFØ145655 54D45Ø1,FFØ17D45
45457DØ1,FFØ179457941 41Ø1,FF
Ø17D45454D7DØ1,FFØ17945794 94
5Ø1
```

LISTINGS

```
19 DATA FF017D417D057D01,FF0
17D111111101,FF01454545457D
01,FF0145452929 1101,FF014545
55557D01,FF0145291129 4501,FF
0145291111 1101,FF017D0911217
D01
20 DATA ,,,,,,FF01311111139
01,FF0139093921 3901,FF013909
19093901,FF0129293909 0901
21 DATA FF0139213909 3901,FF0
1392139293901,FF0139090909 09
01,FF0139091901 1101,FF017D45
7D454501,,,,,,,,00090D0B0909
0057,75555555555557
22 DATA 0070505050700056,545
45654542426,003A2B2A2A3A0067
,55556755555565,005C50D8505C
006A,4A4A4C4C4A4A6A
23 DATA ",******-!",(TL<K:T)
!,(TTTTTT)!,(000000)!,.+++++
+/!,",******-!",(T[|E>T)!,(T
TTTTT)!,(000000)!,.++++++/!,
",******-!",(T1986T)!
24 DATA (TTTTTT)!,(000000)!,
.++++++/!,",******-!",(TTTTT
T)!,(TTTTTT)!,(000000)!,.+++
+++/!,",**-,**-!",(TT)(TT)!,
(TT)(TT)!,.++/.++/#
25 NEXT I1 :: NEXT I :: CALL
 HCHAR(24,10,34,23):: RANDOM
IZE :: FOR I=1 TO 23 :: FOR
I1=1 TO 23 :: CALL H(I,9+I1,
INT(RND*8)+120):: NEXT I1 ::
 NEXT I :: CALL H(12,21,128)
:: FOR I=3 TO 7
26 CALL COLOR(0,5,15,1,7,4,2
,13,4,I,5,15,I+5,7,12,12,7,8
,13,7,16):: NEXT I
27 ER=0 :: CALL SPRITE(#1,36
,14,95,127,#2,136,10,169,9)
28 S$(0)="00090D0B09090057 75
55555555557000070505050 7000
5654545654542426 00" :: S$(1)
="003A2B2A2A3A00675555675555
5565000005C50D8505C006A4A4A4C
4C4A4A6A00"
29 S$(2)="003A2B2A2A3A007545
4575171717770000 05C50D8505C00
76565676545454540 0" :: S$(3)
="007A424343434373424242424 2
427A0000058545252525252D2D2D2
D2D2545800"
30 CALL SPRITE(#3,140,10,169
,41):: FOR I=1 TO 4 :: I2=I*
I+1-2*(I=2 OR I=3):: FOR I1=
1 TO 6 :: W=INT(RND*28)+58 :
: W$(I)=W$(I)&CHR$(W):: CALL
 H(I2+1,I1+1,W)
31 CALL H(I2,I1+1,30):: CALL
 KEY(0,X,Y):: IF Y=0 OR X<97
 OR X>126 OR X=124 THEN 31 E
LSE CALL H(I2,I1+1,X-39):: N
$(I)=N$(I)&CHR$(X-39)
32 NEXT I1 :: NEXT I
33 FOR I=1 TO 4 :: I2=I*I+1-
2*(I=2 OR I=3):: FOR O=1 TO
6 :: CALL CHARPAT(ASC(SEG$(N
$(I),O,1)),I$):: CALL CHAR(1
29+O,I$):: NEXT O :: CALL H(
I2,2,130):: DISPLAY AT(I2,1)
:"";:: S=1
34 ZUG=0 :: ZU=1-1*(I=2 OR I
=4):: Z=I2+1 :: CALL LOCATE(
#1,Z*8-7,S*8-7):: IF N$(I)="
TTTTTT" THEN 46
35 CALL JOYST(ZU,X,Y):: IF X
=4 THEN S=S-1*(S<31):: GOTO
38 ELSE IF X=-4 THEN S=S+1*(
S>1):: GOTO 38 ELSE IF Y=4 T
HEN Z=Z+1*(Z>1):: GOTO 38 EL
SE IF Y=-4 THEN Z=Z-1*(Z<23)
:: GOTO 38 ELSE CALL KEY(ZU,
X,Y)
36 IF X=10 THEN 44 ELSE CALL
 COINC(#1,#2,10,C1):: CALL C
OINC(#1,#3,10,C2):: CALL COI
NC(#1,#4,10,C3):: CALL COINC
(#1,#5,10,C4):: IF (C1 OR C2
 OR C3 OR C4)=0 AND X=18 THE
N 39
37 IF C1=-1 AND X=18 AND T=0
 THEN 46 ELSE IF C2=-1 AND X
=18 AND T=0 THEN 47 ELSE IF
C3=-1 AND X=18 AND T=0 THEN
48 ELSE IF C4=-1 AND X=18 AN
D T=0 THEN 52 ELSE 35
38 CALL LOCATE(#1,Z*8-7,S*8-
7):: GOTO 35
39 IF T=0 AND Z=I2+1 AND S<7
 AND ZUG<6 THEN 40 ELSE IF T
=1 AND S>8 THEN CALL GCHAR(Z
,S+1,G):: IF G=128 AND ER=0
THEN ER=1 :: GOTO 41 ELSE IF
 G>119 AND G<128 AND ER=1 TH
EN 41 ELSE 35 ELSE 35
40 CALL GCHAR(Z,S+1,G):: IF
G=84 THEN 35 ELSE L1=G :: Z1
=Z :: S1=S :: ZUG=ZUG-1*(ZUG
<6):: CALL CHARPAT(G,I$):: C
ALL CHAR(129,I$):: CALL H(Z,
S+1,129):: T=1 :: CALL C(1,1
6):: GOTO 35
41 L2=G :: T,TY=0 :: Z2=Z ::
 S2=S :: CALL H(Z1,S1+1,INT(
RND*28)+58):: IF G=127 THEN
K1=INT(RND*101):: GOTO 42 EL
SE K1=10*(G-119)*-(G<>128)
42 CALL P(P(),I,K1):: CALL C
(1,14):: IF L1=85 THEN 43 EL
SE CALL H(Z,S+1,L1+30):: GOT
O 35
```

```
43 CALL KEY(Ø,X,Y):: IF Y=Ø
OR X<97 OR X>122 THEN 43 ELS
E CALL H(Z,S+1,X-9):: GOTO 3
5
44 IF J=Ø THEN CALL DELSPRIT
E(#2,#3):: CALL CHAR(136,S$(
2),14Ø,S$(3)):: CALL SPRITE(
#4,136,1Ø,169,9,#5,14Ø,1Ø,16
9,41):: J=1 :: GOTO 35
45 CALL DELSPRITE(#4,#5):: C
ALL CHAR(136,S$(Ø),14Ø,S$(1)
):: CALL SPRITE(#2,136,1Ø,16
9,9,#3,14Ø,1Ø,169,41):: J=Ø
:: GOTO 35
46 CALL S :: CALL C(3,1):: F
OR IØ=1 TO 6 :: CALL H(I2,IØ
+1,ASC(SEG$(N$(I),IØ,1))):: 
NEXT IØ :: CALL C(3,1Ø):: T,
TU,ZUG=Ø :: NEXT I :: GOTO 3
3
47 IF TY=1 AND ZUG<=Ø THEN 3
5 ELSE CALL S :: CALL C(2,1)
:: CALL H(Z2,S2+1,L2):: CALL
 H(Z1,S1,L1):: CALL P(P(),I,
-K1):: TY=1 :: ZUG=ZUG-1 ::
CALL C(2,1Ø):: GOTO 35
48 IF TU=6 THEN 35 ELSE CALL
 S :: CALL C(5,1):: H1=I*I+2
-2*(I=2 OR I=3):: S6=1 :: CA
LL LOCATE(#1,H1*8-7,S6*8-7)
49 CALL JOYST(1,X,Y):: IF X=
4 THEN S6=S6-1*(S6<6):: GOTO
 5Ø ELSE IF X=-4 THEN S6=S6+
1*(S6>1):: GOTO 5Ø ELSE CALL
 KEY(1,X,Y):: IF X=18 THEN 5
1 ELSE 49
5Ø CALL LOCATE(#1,H1*8-7,S6*
8-7):: GOTO 49
51 CALL H(H1,S6+1,INT(RND*28
)+58):: TU=TU+1 :: CALL C(5,
1Ø):: GOTO 35
52 CALL S :: CALL C(4,1):: F
OR I=1 TO 3 :: FOR I1=4 TO I
 STEP -1 :: IF P(I)<P(I1)THE
N J1=P(I1):: P(I1)=P(I):: P(
I)=J1 :: J1$=N$(I1):: N$(I1)
=N$(I):: N$(I)=J1$
53 NEXT I1 :: NEXT I :: F$(1
)="?BKLMT" :: F$(2)="L><HG="
 :: F$(3)="MABK=T" :: F$(4)=
"?HNKMA" :: FOR I=1 TO 4 ::
I2=I*I+1-2*(I=2 OR I=3):: FO
R I1=1 TO 6
54 CALL H(I2,I1+1,ASC(SEG$(F
$(I),I1,1))):: CALL H(I2+1,I
1+1,ASC(SEG$(N$(I),I1,1))):: 
 CALL H(I2+2,I1+1,ASC(SEG$(R
PT$("Ø",6-LEN(STR$(P(I))))&S
TR$(P(I)),I1,1))):: NEXT I1
:: NEXT I
55 FOR I=1 TO 1ØØØ :: CALL S
 :: CALL SOUND(-1ØØ,-3,Ø)::
CALL SOUND(-1ØØ,-5,Ø):: CALL
 KEY(Ø,X,Y):: IF X=32 THEN 5
7 ELSE 56
56 NEXT I
57 CALL DELSPRITE(ALL):: CAL
L CLEAR :: END
58 SUB P(P(),I,A):: P(I)=P(I
)+A :: A$=RPT$("Ø",6-LEN(STR
$(P(I))))&STR$(P(I)):: FOR I
6=1 TO 6 :: CALL H(I*I+3-2*(
I=2 OR I=3),I6+1,ASC(SEG$(A$
,I6,1))):: NEXT I6 :: SUBEND
59 SUB C(A,B):: CALL COLOR(#
A,B):: SUBEND
6Ø SUB H(A,B,C):: CALL HCHAR
(A,B,C):: SUBEND
61 SUB S :: CALL SOUND(-1ØØ,
-1,Ø):: SUBEND
```

Ram-Anzeige

Diese Pokeliste ist eine Unterstützung für jeden Basic-Programmierer.
Dem User wird dabei ständig der aktuelle Stand des Speicherraumes angezeigt. Somit hat man stets einen Überblick über die Größe des Programms.
Gestartet wird diese Routine über CALL LINK ("HEAD").
Gleich darauf erscheinen in der ersten Zeile die noch freien Speicherplätze.
Selbst wenn ein Basicprogramm gestartet wird, läuft diese Routine weiter.
Durch CALL LINK("STOP") läßt sich das Programm ausschalten.

```
100 ! *********************
110 ! *                   *
120 ! * RAM-Basic-Anzeige *
130 ! *                   *
140 ! *    Copyright by   *
150 ! *  Wolfgang Sommer  *
160 ! *                   *
170 ! *                   *
180 ! *                   *
190 ! * Benoetigte Geraete *
200 ! *  TI99/4A  Konsole *
210 ! *     Ext. Basic    *
220 ! *        32K        *
230 ! *                   *
240 ! *                   *
250 ! *                   *
260 ! *  Speicherbelegung *
270 ! *     2429 Bytes    *
280 ! *                   *
290 ! *********************
300 CALL INIT
310 ADR=9460
320 READ A :: IF A<>-1 THEN
```

```
CALL LOAD(ADR,A):: ADR=ADR+1
 :: GOTO 320
330 CALL LOAD(8194,36,244,63
,240)
340 FOR I=16368 TO 16383 ::
READ A :: CALL LOAD(I,A):: N
EXT I
350 DATA 000,005,146,000,000
,032,000,000,000,000,000,000
,203,020,203,053
360 DATA 203,078,203,231,204
,071,204,150,204,228,205,029
,205,075,205,096
370 DATA 160,063,000,010,128
,128,128,128,128,128,128,128
,128,128,128,128
380 DATA 128,162,185,180,165
,135,179,128,178,161,173,128
,166,178,165,169
390 DATA 128,128,128,128,002
,001,037,068,200,001,131,196
,004,096,000,112
400 DATA 002,224,036,244,004
,192,002,001,037,024,002,002
,000,032,004,032
410 DATA 032,036,002,000,000
,010,193,032,131,135,097,032
,037,020,004,195
420 DATA 060,224,037,022,192
,068,006,193,002,033,144,000
,004,032,032,032
430 DATA 006,000,193,003,022
,244,002,224,131,224,004,091
,004,224,131,196
440 DATA 004,096,000,112,084
,207,076,069,078,213,076,069
,084,141,076,079
450 DATA 071,208,077,065,088
,223,077,073,078,224,078,069
,087,000,078,079
460 DATA 084,189,078,085,077
,004,079,076,068,005,080,079
,083,217,082,069
470 DATA 067,222,082,069,077
,154,082,069,083,006,082,078
,068,215,082,085
480 DATA 078,169,083,071,078
,209,083,073,078,210,083,081
,082,211,083,085
490 DATA 066,161,084,065,066
,252,084,065,078,212,086,065
,076,218,088,079
500 DATA -1
510 REM NAMENSTABELLE:
520 DATA 083,084,079,080,032
,032,037,128,072,069,065,068
,032,032,037,056
```

Desktop

„Desktop Fighters“ ist ein Spiel für zwei Spieler, von denen jeder das Kommando über eine Flotte von Raumschiffen besitzt. Das Ziel des Spiels ist es, alle fünfzehn Sternensysteme in diesem Sektor zu besetzen. Dazu können die Flotten beliebig im Raum verschoben und aufgeteilt werden.
Nach dem Start des Programms erscheint nach einiger Zeit die Frage, ob man ein neues Spiel anfangen oder einen abgespeicherten Spielstand laden will. In diesem Menü kann wie in allen anderen mit den Tasten S und D ein Pfeil neben den Menüpunkten bewegt werden. Mit ENTER wählt man den Punkt an, neben dem der Pfeil steht.
Nachdem man „Neues Spiel“ angewählt hat, landet man nach einiger Zeit im Hauptmenü. Hier kann man wählen zwischen:

a) Karte

Mit diesem Punkt kann man sich eine strategische Karte zeigen lassen, auf der die Flotten der Spieler und die Systeme angezeigt werden. Rechts oben auf dem Schirm steht das Symbol für eine Flotte des entsprechenden Spielers.

b) Flotten bewegen

Nach der Auswahl dieses Punktes wird zuerst eine Karte gezeigt, auf der man mit einem durch ESDXWZRC gesteuerten Fadenkreuz eine eigene Flotte anwählen (zielen und ENTER). Mit CTRL und einem Buchstaben von A–P kann man eine bestimmte Flotte direkt anspringen. Flotten, die bereits einen Kurs haben, werden als weiße Kreise dargestellt.
Nach ein paar Sekunden erscheint ein blinkender Cursor auf der Flotte. Man kann nun die Flotte wie das Fadenkreuz steuern, allerdings nur eine bestimmte Zahl von Schritten, die von der Geschwindigkeit abhängt (siehe „Schiffsbuch“). Mit ENTER beendet man die Kursprogrammierung.

c) Flotten teilen

Mit dieser Option kann man eine Flotte in zwei Flotten aufspalten. Wieder wird wie bei b) eine Flotte gewählt. Danach öffnet sich ein Fenster, in dem die Zusammensetzung der Flotte nach Typen und Anzahl angezeigt wird. Mit dem bekannten Pfeil kann der Spieler einen Schiffstyp anwählen und nach ENTER die Anzahl der Schiffe dieses Typs angeben, die in der neuen Flotte sein sollen. Durch Anwahl von „Fertig“ wird die Teilung beendet.
Die neue Flotte erscheint auf einem der Nachbarfelder. Wenn kein angrenzendes Feld mehr frei ist oder ein Spieler schon 16 Flotten hat, ist eine Flottenteilung nicht mehr möglich.

Fighters

d) Statusreport

Nach der Auswahl dieses Punktes öffnet sich ein Untermenü, in dem man die Auswahl hat zwischen

1) **S tand:** gibt die Anzahl der Flotten und besetzte Systeme jedes Spielers an. Achtung: Die Auszählung kann bis zu ca. 10 Sekunden dauern.
2) **F lotteninfos:** präsentiert zuerst das bekannte Fadenkreuz, mit dem man wie bei b) eine Flotte wählt. Danach wird in einem Fenster die Zusammensetzung der Flotte angezeigt (nach Schiffstypen).
3) **K ommando:** bringt einen wieder ins Hauptmenü.

e) Schiffsbuch

Dieser Programmteil ist ein besonderes Bonbon. Hier kann man sich die Daten über die im Spiel verwendeten Schiffstypen geben lassen. Zu jedem Typ werden die Faktoren Geschwindigkeit, Panzerung und Bewaffnung als Balken zusammen mit seiner Funktion und einem kleinen Bild des Schiffs angezeigt.
Mit S und D kann man durch die Liste blättern, mit ENTER geht's zurück ins Hauptmenü.

f) Zug beenden

Die Anwahl dieser Option beendet die Zugeingabe und führt die Flugbefehle für die Flotten aus. Wenn dabei zwei eigene Flotten *nach* Ende ihres Zuges auf dem gleichen Feld stehen, werden sie zu einer Flotte vereinigt. Kommt eine Flotte in einen Sektor, in dem sich schon eine gegnerische befindet, so beginnt ein Kampf. Als erstes muß man in der üblichen Menütechnik für jedes Schiff die Angriffsziele wählen.
Danach kann man sich die Strategie für den Angriff aussuchen. „Massiver Angriff" und „Schlag-zu-und-hau-ab" sprechen wohl für sich. Mit „Dogfight" ist ein Luftkampf Schiff gegen Schiff gemeint. Es ist bei der Programmierung der Ziele und der Auswahl der Strategie natürlich zu beachten, daß nicht jeder Schiffstyp für jede Strategie gleich gut geeignet ist („Dogfight" mit schweren Schiffen ist der sicherste Weg, die Schlacht zu verlieren).
Im sich nun öffnenden Fenster wird ständig Bericht über die Kampfhandlungen gegeben. Ist eine Flotte völlig vernichtet, geht die Flugphase mit den anderen Flotten weiter.
Nachdem alle Flotten bewegt worden sind, kann das Spiel entweder abgespeichert oder fortgesetzt werden. Wenn man weiterspielen will, geht es für den anderen Spieler wieder im Hauptmenü los. Tritt beim Abspeichern ein Fehler auf, so wird nicht abgespeichert, doch das Programm wird ohne Datenverlust fortgesetzt.
Einen abgespeicherten Spielstand kann man beim Programmstart wieder einladen.

Ein Tip: Mir persönlich macht es mehr Spaß, so zu spielen, daß kein Spieler den Zug und insbesondere die Flottenstruktur des anderen beobachten kann.
Viel Spaß!

Marcus Gröber

```
100 ! **********************
110 ! *                    *
120 ! *      DESKTOP       *
130 ! *        FIGHTERS    *
140 ! *    Copyright by    *
150 ! *   Marcus Groeber   *
160 ! *                    *
170 ! *                    *
180 ! *                    *
190 ! * Benoetigte Geraete *
200 ! *  TI99/4A  Konsole  *
210 ! *      Ext. Basic    *
220 ! * Kassettenrecorder  *
230 ! *                    *
240 ! *                    *
250 ! *                    *
260 ! *  Speicherbelegung  *
270 ! *     11220 Bytes    *
280 ! *                    *
290 ! **********************
300 CALL CLEAR :: RANDOMIZE
:: CALL SCREEN(5):: CALL MAG
NIFY(3)
310 DIM SY(14,1),FL(1,15,1),
FS$(1,15),FC$(15),FC(1),PC(1
)
320 FOR A=40 TO 90 :: CALL C
HARPAT(A,A$):: CALL CHAR(A,S
EG$(A$,3,14)):: NEXT A
330 FOR A=97 TO 122 :: CALL
CHARPAT(A,A$):: CALL CHAR(A,
SEG$(A$,5,12)&"00FF"):: NEXT
 A
340 CALL CHAR(123,"80BFBFBFB
FBF80BFBFBFBFBF80E0E0FF03FBF
BF9F9F901F9F9F9F9F9010101FF"
,127,"FF00FF00FF00FF")
350 CALL CHAR(128,"FF0000000
000FF0000000000FFFFFF0040404
04040404040"&RPT$("06",8))
360 CALL CHAR(64,RPT$("F",16
),38,"FFB7B7B7B7B7FF",39,"FF
EDEDEDEDEDFF",96,RPT$("0",14
)&"FF",140,"00060F4F06008004
")
370 CALL CHAR(35,"00FF550101
55FF00207EFE7E2000000008FCFE
FC08",136,"030D112140408OF08
080404021110D03C030080402020
10F0101020204083OC")
380 CALL CHAR(42,"FFF7D5E380
E3D5F7",59,"FFFFE7C3C3E7FFFF
FFFDE18D01E1FFFFFFBF87B18087
FFFFEFBDE18D01E1BFFD",63,"EF
BD87B18087FDDF")
390 @$=RPT$(CHR$(0),8):: CAL
L HCHAR(2,1,127,736)
400 FOR A=0 TO 13 :: CALL CO
LOR(A,2,15):: NEXT A :: FOR
A=9 TO 12 :: CALL COLOR(A,15
,2):: NEXT A :: CALL COLOR(1
```

LISTINGS

```
4,11,2)
410 CALL YN("NEUES SPIEL","A
LTES SPIEL",A):: IF A=1 THEN
 PL=1 :: GOTO 450
420 CALL CLEAR :: OPEN #1:"C
S1",INTERNAL,INPUT ,FIXED 12
8 :: PRINT :"* READING"|:: I
NPUT #1:PL
430 FOR B=0 TO 15 :: IF B<15
 THEN INPUT #1:SY(B,0),SY(B,
1),
440 INPUT #1:FL(0,B,0),FL(0,
B,1),FL(1,B,0),FL(1,B,1),FS$
(0,B),FS$(1,B):: NEXT B :: C
LOSE #1 :: GOTO 490
450 FOR PL=0 TO 14
460 PX=INT(RND*26+1):: PY=IN
T(RND*20+1):: FOR A=1 TO PL-
1 :: IF SY(A,0)=PY AND SY(A,
1)=PX THEN 460
470 NEXT A :: SY(PL,0)=PY ::
 SY(PL,1)=PX :: NEXT PL
480 FOR PL=0 TO 1 :: FOR A=0
 TO 1 :: FS$(PL,A)="K}ZnP2t<
" :: FL(PL,A,1)=PL*25+1 :: F
L(PL,A,0)=A*19+1 :: NEXT A :
: NEXT PL :: PL=1
490 !ON ERROR 1500
500 PL=1-PL :: DISPLAY AT(1,
1)SIZE(28):"SPIELER ";CHR$(6
5+PL);TAB(26);CHR$(60+PL)
510 QY=9 :: QX=13 :: NC=-1 :
: CALL HCHAR(2,1,127,736)
520 CALL WIN("&kommandos'","
 ",5,7,20,23):: CALL D(7,9,"
FLOTTENKURSE"):: CALL D(9,9,
"FLOTTEN TEILEN")
530 CALL D(11,9,"STATUSREPOR
T"):: CALL D(13,9,"KARTE")::
 CALL D(15,9,"SCHIFFSBUCH")
540 CALL D(17,9,"ZUG BEENDEN
"):: CALL CHO(7,8,6,CH,19,10
)
550 IF CH<6 THEN ON CH GOSUB
 1270,1040,1180,1030,1390 ::
 GOTO 520
560 IF NOT NC THEN TT$="flug
" :: GOSUB 1510
570 FOR FP=0 TO 15 :: G=59
580 IF FS$(PL,FP)="" THEN 93
0
590 IF SEG$(FS$(PL,FP),9,1)=
"*" THEN FS$(PL,FP)=SEG$(FS$
(PL,FP),1,8)
600 IF NC OR FC$(FP)="" THEN
 930
610 A$=FC$(FP):: K=ASC(SEG$(
A$,LEN(A$),1)):: FC$(FP)=SEG
$(A$,1,LEN(A$)-1):: GOSUB 16
90
620 Y1=FL(PL,FP,0)+Y :: X1=F
L(PL,FP,1)+X :: CALL GCHAR(Y
1+3,X1+3,A):: CALL VCHAR(Y1+
3,X1+3,60+PL-2*(A>139)):: CA
LL VCHAR(Y1-Y+3,X1-X+3,G)
630 FL(PL,FP,0)=Y1 :: FL(PL,
FP,1)=X1 :: G=A :: IF G=64 T
HEN 600
640 FOR A=0 TO 15 :: IF FL(P
L,A,0)=Y1 AND FL(PL,A,1)=X1
AND A<>FP AND FC$(A)="" AND
FC$(FP)="" AND FS$(PL,A)>""
THEN 680
650 IF FL(1-PL,A,0)=Y1 AND F
L(1-PL,A,1)=X1 AND FS$(1-PL,
A)>"" THEN 700
660 NEXT A
670 GOTO 600
680 FOR Y=1 TO 8 :: CALL I(F
S$(PL,A),ASC(SEG$(FS$(PL,A),
Y,1))+ASC(SEG$(FS$(PL,FP),Y,
1)),Y):: NEXT Y
690 FS$(PL,FP)="" :: GOTO 93
0
700 X=1-PL :: CALL WIN("&ala
rm'"," ",5,4,21,25):: CALL M
(7,"FEINDSCHIFFE IN"):: CALL
 M(8,"REICHWEITE!")
710 CALL M(9,"ZIELE WAEHLEN!
"):: A$=FS$(PL,FP):: B$=FS$(
X,A):: C$="" :: FOR Y=1 TO 8
720 IF ASC(SEG$(B$,Y,1))THEN
 C$=C$&CHR$(Y):: CALL S(10+L
EN(C$),17,Y)
730 NEXT Y :: CALL D(12,8,"t
yp"):: CALL D(15,7,"anzahl")
:: TA$=""
740 FOR Y=1 TO 8 :: F=ASC(SE
G$(A$,Y,1)):: IF F=0 THEN CH
=0 :: GOTO 760 ELSE CALL S(1
3,6,Y):: CALL D(16,8,STR$(F)
&"  ")
750 CALL CHO(11,16,-LEN(C$),
C2,20,9):: CH=ASC(SEG$(C$,C2
,1))
760 TA$=TA$&CHR$(CH):: NEXT
Y :: CALL WIN("&strategie'",
" ",8,2,15,27)
770 CALL D(10,4,"MASSIVER AN
GRIFF"):: CALL D(11,4,"""SCH
LAG ZU UND HAU AB"""):: CALL
 D(12,4,"""DOGFIGHT""")
780 CALL CHO(10,3,-3,A,14,9)
:: IF A=1 THEN P1=1 :: P2=9
:: P3=7 ELSE IF A=2 THEN P1=
5 :: P2=9 :: P3=5 ELSE P1=9
:: P2=5 :: P3=2
790 CALL WIN("&kampfbericht'
"," ",6,3,19,26)
800 CALL D(8,16,"'"&CHR$(PL+
97)&"'     '"&CHR$(X+97)&"'"):
```

```
: FOR Y=1 TO 8 :: CALL S(8+Y
,5,Y):: NEXT Y
810 FOR Y=1 TO 8 :: DISPLAY
AT(8+Y,16)SIZE(9):USING "###
   ###":ASC(SEG$(A$,Y,1)),AS
C(SEG$(B$,Y,1)):: NEXT Y ::
IF A$=@$ OR B$=@$ THEN 890
820 FOR Y=1 TO 8 :: IF A$=@$
 OR B$=@$ THEN 810
830 X=ASC(SEG$(TA$,Y,1)):: N
1=ASC(SEG$(A$,Y,1)):: IF X=0
 OR N1=0 THEN 880 ELSE N2=AS
C(SEG$(B$,X,1)):: IF N2 THEN
 850
840 X=INT(RND*8+1):: N2=ASC(
SEG$(B$,X,1)):: IF N2=0 THEN
 840 ELSE CALL I(TA$,X,Y)
850 CALL P(Y,P1,P2,P3):: CAL
L P(X,D1,D2,D3)
860 L2=P1*F1+P2*F2+P3*F3 ::
L1=D1*F1+D2*F2+D3*F3 :: L2=L
2*L2*L2/(5E5*(RND+2.5)):: L1
=L1*L1*L1/(5E5*(RND+3))
870 CALL I(B$,MAX(0,N2-L2*N1
),X):: CALL I(A$,MAX(0,N1-L1
*N2),Y)
880 NEXT Y :: GOTO 810
890 IF A$=@$ THEN A$=""
900 IF B$=@$ THEN B$=""
910 FS$(PL,FP)=A$ :: FS$(1-P
L,A)=B$ :: FC$(FP)=""
920 CALL WKEY(18,19):: GOSUB
 1510 :: GOTO 600
930 NEXT FP
940 GOSUB 1650 :: IF PC(PL)<
15 AND FC(1-PL)>0 THEN 980
950 A$=CHR$(65+PL):: CALL WI
N("&spielende'"," ",8,3,15,2
6):: CALL M(10,"SPIELER "&A$
&" HAT")
960 CALL M(12,"GEWONNEN!")::
 CALL WKEY(14,11)
970 CALL CLEAR :: END
980 IF FC(PL)=0 THEN PL=1-PL
 :: GOTO 950
990 CALL YN("WEITER","ABSPEI
CHERN",A):: IF A=1 THEN 500
1000 CALL CLEAR :: OPEN #1:"
CS1",INTERNAL,OUTPUT,FIXED 1
28 :: PRINT :"* WRITING";::
PRINT #1:PL
1010 FOR B=0 TO 15 :: IF B<1
5 THEN PRINT #1:SY(B,0),SY(B
,1),
1020 PRINT #1:FL(0,B,0),FL(0
,B,1),FL(1,B,0),FL(1,B,1),FS
$(0,B),FS$(1,B):: NEXT B ::
CLOSE #1 :: GOTO 500
1030 TT$="karte" :: GOSUB 15
10 :: CALL WKEY(24,11):: RET
```

LISTINGS

```
URN
1040 FOR F=0 TO 15 :: IF FS$
(PL,F)="" THEN GOSUB 1570 ::
 IF A=-1 THEN RETURN ELSE IF
 SEG$(FS$(PL,A),9,1)="*" THE
N 1170 ELSE 1060
1050 NEXT F :: GOTO 1170
1060 FOR Y1=QY-1 TO QY+1 ::
FOR X1=QX-1 TO QX+1 :: IF X1
>0 AND Y1>0 AND X1<27 AND Y1
<21 THEN 1080
1070 NEXT X1 :: NEXT Y1 :: G
OTO 1170
1080 FOR FP=0 TO 1 :: FOR K=
0 TO 15 :: IF FL(FP,K,0)=Y1
AND FL(FP,K,1)=X1 AND FS$(FP
,A)>"" THEN 1070
1090 NEXT K :: NEXT FP
1100 CALL WIN("&flotte'teile
n'"," ",6,3,19,26):: C$="###
 0" :: GOSUB 1380
1110 CALL D(16,5,"fertig")::
 B$=@$ :: C$=FS$(PL,A)
1120 CALL CHO(8,4,-9,CH,18,9
):: IF CH=9 THEN 1150
1130 ACCEPT AT(7+CH,22)SIZE(
-3)VALIDATE(DIGIT):A$ :: X=V
AL("0"&A$):: V=ASC(SEG$(FS$(
PL,A),CH,1)):: IF X>V THEN 1
130
1140 CALL I(B$,X,CH):: CALL
I(C$,V-X,CH):: DISPLAY AT(7+
CH,18)SIZE(3):USING "###":V-
X :: GOTO 1120
1150 IF B$=@$ THEN RETURN EL
SE FS$(PL,F)=B$&"*" :: FL(PL
,F,0)=Y1 :: FL(PL,F,1)=X1 ::
 IF C$=@$ THEN C$=""
1160 FS$(PL,A)=C$ :: RETURN
1170 CALL ER("FLOTTENTEILUNG
","NICHT MOEGLICH!"):: RETUR
N
1180 CALL WIN("&statusreport
'"," ",7,4,16,25)
1190 CALL D(9,6,"STAND"):: C
ALL D(11,6,"FLOTTENINFOS")::
 CALL D(13,6,"KOMMANDO")
1200 CALL CHO(9,5,3,CH,15,11
):: IF CH=3 THEN RETURN
1210 IF CH=2 THEN 1250
1220 CALL WIN("stand"," ",15
,13,22,26):: GOSUB 1650 :: C
ALL D(17,14,"FLOTTEN:a'"&STR
$(FC(0)))
1230 CALL D(18,22,"b'"&STR$(
FC(1))):: CALL D(19,14,"SYST
EME:a'"&STR$(PC(0))):: CALL
D(20,22,"b'"&STR$(PC(1)))
1240 CALL WKEY(21,14):: GOTO
 1180
1250 GOSUB 1570 :: IF A=-1 T
```

```
HEN 1180 ELSE CALL WIN("info
s'flotte'"&CHR$(A+97)," ",5,
3,17,26)
1260 CALL D(7,7,"typ      an
zahl"):: C$="###" :: GOSUB 1
380 :: CALL WKEY(16,18):: GO
TO 1180
1270 DM=1 :: GOSUB 1570 :: I
F A=-1 THEN RETURN
1280 F=A :: SP=9 :: FOR A=1
TO 8 :: CALL P(A,F1,F2,F3)::
 IF ASC(SEG$(FS$(PL,F),A,1))
THEN SP=MIN(SP,F1)
1290 NEXT A :: DISPLAY AT(3,
6)SIZE(18):"kurs'programmier
en"
1300 A$="*" :: B$="" :: PY=F
L(PL,F,0)+3 :: PX=FL(PL,F,1)
+3
1310 CALL GCHAR(PY,PX,G):: B
=G+42
1320 G=B-G :: CALL VCHAR(PY,
PX,G):: FOR A=1 TO 10 :: CAL
L KEY(3,K,S):: IF S=1 THEN 1
340
1330 NEXT A :: GOTO 1320
1340 IF K=13 THEN CALL T ::
CALL VCHAR(PY,PX,42):: FC$(F
)=SEG$(A$,1,LEN(A$)-1):: NC=
0 :: RETURN ELSE GOSUB 1690
:: IF X=0 AND Y=0 THEN 1320
1350 X1=PX+X :: Y1=PY+Y :: I
F X1<4 OR X>29 OR Y1<4 OR Y1
>23 THEN 1320
1360 A=K :: K=ASC(A$):: GOSU
B 1690 :: IF X1+X=PX AND Y1+
Y=PY THEN CALL VCHAR(PY,PX,B
-42):: G=ASC(B$):: B=G+42 ::
 A$=SEG$(A$,2,9):: B$=SEG$(B
$,2,9):: PX=X1 :: PY=Y1 :: G
OTO 1320
1370 IF LEN(B$)<SP THEN CALL
 VCHAR(PY,PX,59):: PX=X1 ::
PY=Y1 :: A$=CHR$(A)&A$ :: B$
=CHR$(B-42)&B$ :: GOTO 1310
ELSE 1320
1380 FOR Y=1 TO 8 :: CALL S(
7+Y,5,Y):: DISPLAY AT(7+Y,18
)SIZE(LEN(C$)):USING C$:ASC(
SEG$(FS$(PL,A),Y,1)):: NEXT
Y :: RETURN
1390 SH=1 :: RESTORE 1710 ::
 CALL WIN("&schiffsbuch'","
",14,2,22,27)
1400 CALL D(16,3,"typ:"):: C
ALL D(18,7,"MAX.GESCHW.")::
CALL D(19,7,"BEWAFFNUNG")
1410 CALL D(20,7,"PANZERUNG"
):: CALL WIN(""," ",18,3,21,
6):: CALL D(21,11,"$S D% ent
er ENDE")
1420 READ B$,CH$ :: CALL CHA
R(92,CH$):: CALL P(SH,F1,F2,
F3):: CALL S(16,7,SH):: DISP
LAY AT(17,3)SIZE(24):B$
1430 CALL D(19,4,"\^"):: CAL
L D(20,4,"]_")
1440 DISPLAY AT(18,18)SIZE(9
):RPT$("#",F1):: DISPLAY AT(
19,18)SIZE(9):RPT$("#",F2)::
 DISPLAY AT(20,18)SIZE(9):RP
T$("#",F3)
1450 CALL KEY(3,K,S):: IF S=
0 THEN 1450
1460 IF K=68 AND SH<8 THEN S
H=SH+1 :: GOTO 1420 ELSE IF
K=68 THEN SH=1 :: RESTORE ::
 GOTO 1420
1470 IF K<>83 THEN 1500
1480 IF SH>1 THEN SH=SH-1 EL
SE SH=8
1490 RESTORE 1710 :: FOR A=2
 TO SH :: READ B$,CH$ :: NEX
T A :: GOTO 1420
1500 IF K<>13 THEN 1450 ELSE
 RETURN
1510 CALL WIN(TT$,"@",3,1,24
,28)
1520 FOR A=0 TO 14 :: CALL V
CHAR(SY(A,0)+3,SY(A,1)+3,140
):: NEXT A
1530 FOR X1=0 TO 1 :: FOR A=
0 TO 15 :: IF FS$(X1,A)="" T
HEN 1560
1540 Y=FL(X1,A,0)+3 :: X=FL(
X1,A,1)+3 :: IF DM AND X1=PL
 AND FC$(A)>"" THEN B=42 ELS
E CALL GCHAR(Y,X,G):: B=60+X
1-2*(G=140)
1550 CALL VCHAR(Y,X,B)
1560 NEXT A :: NEXT X1 :: DM
=0 :: RETURN
1570 TT$="&flotte'waehlen'"
:: GOSUB 1510
1580 CALL SPRITE(#1,136,8,13
+QY*8,13+QX*8)
1590 CALL KEY(5,K,S):: IF S=
0 THEN 1590 ELSE IF K=13 THE
N CALL DELSPRITE(#1):: CALL
T :: GOTO 1630
1600 IF K>128 AND K<145 THEN
 A=K-129 :: IF FS$(PL,A)>""
THEN QX=FL(PL,A,1):: QY=FL(P
L,A,0):: CALL LOCATE(#1,13+Q
Y*8,13+QX*8):: GOTO 1590
1610 GOSUB 1690 :: IF X=0 AN
D Y=0 THEN 1590
1620 IF QX+X<1 OR QX+X>26 OR
 QY+Y<1 OR QY+Y>20 THEN 1580
 ELSE QX=QX+X :: QY=QY+Y ::
GOTO 1580
```

```
1630 FOR A=0 TO 15 :: IF FL(
PL,A,0)=QY AND FL(PL,A,1)=QX
 AND FS$(PL,A)>"" THEN RETUR
N
1640 NEXT A :: A=-1 :: CALL
ER("IN DIESEM SEKTOR IST","K
EINE IHRER FLOTTEN!"):: RETU
RN
1650 FOR FP=0 TO 1 :: FC(FP)
,PC(FP)=0 :: FOR F=0 TO 15
1660 IF FS$(FP,F)>"" THEN FC
(FP)=FC(FP)+1 ELSE 1680
1670 FOR A=0 TO 14 :: PC(FP)
=PC(FP)-(SY(A,0)=FL(FP,F,0)A
ND SY(A,1)=FL(FP,F,1)):: NEX
T A
1680 NEXT F :: NEXT FP :: RE
TURN
1690 Y=(K=87 OR K=69 OR K=82
)-(K=90 OR K=88 OR K=67):: X
=(K=87 OR K=83 OR K=90)-(K=8
2 OR K=68 OR K=67):: RETURN
1700 ON ERROR 1700 :: RETURN
 500
1710 DATA SCHWERER RAUMFRACH
TER,0000000000000003050A182727
1F070100000000063EFE0F214393
278EFC98906
1720 DATA KLEINER JAEGER,00C
0606070303F2A557FC6000000000
0000000000000C038FF8
1730 DATA RUBINLASERKANONE,0
81C0804030508OC0A3B47FF45390
F0000041A26CC78B0F0F0F0F8FCF
8F0C
1740 DATA KAMPFSATELLIT,0000
100B060F101110160D0B17182000
000106DA243C3878F8F8F8F8C402
1750 DATA RAUMAUFKLAERER,000
000000C1277FFFF7F1E7C6040000
000307F3F1D3FD1F7F7F73F113F7
F6
1760 DATA SCHWERER KREUZER,0
00000000E172B377F7FFFFFFC7C1
F0F00000000F8C0C0F0F8FEFFF1F
EF8F0C
1770 DATA BEGLEITSCHIFF,0000
037E39498B40310F030101010200
0000C060FCFEFFBE5CF0C0808080
4
1780 DATA KURIERSCHIFF,805F2
A747A7D6E47030103130B0703030
00000100804BF41FFD5EABCF8D0E
0C
1790 SUB S(Y,X,S):: CALL D(Y
,X,SEG$("RX-192  HORNET  T-1
8    STINGER V-7B    COSMO-7
 VEGA     FIREBIRD",S*8-7,8))
:: SUBEND
1800 SUB P(N,A,B,C):: A=VAL(
SEG$("38157279",N,1)):: B=VA
L(SEG$("35863954",N,1)):: C=
VAL(SEG$("82615934",N,1))::
SUBEND
1810 SUB WIN(N$,S$,Y,X,Y2,X2
):: WW=X2-X+1 :: WB$=CHR$(12
4)&RPT$(CHR$(129),WW-2)&"~"
1820 L1=INT((WW-2-LEN(N$))/2
):: WT$="{"&RPT$(CHR$(128),L
1)&N$&RPT$(CHR$(128),WW-2-LE
N(N$)-L1)&"}"
1830 WS$=CHR$(130)&RPT$(S$,W
W-2)&CHR$(131):: DISPLAY AT(
Y,X)SIZE(WW):WT$
1840 FOR A=Y+1 TO Y2-1 :: DI
SPLAY AT(A,X)SIZE(WW):WS$ ::
 DISPLAY AT(A+1,X)SIZE(WW):W
B$ :: NEXT A
1850 SUBEND
1860 SUB CHO(Y,X,N,C,A,B)::
CALL D(A,B,"$S D% enter")::
A=2+(N<0):: N=ABS(N):: Z=Y :
: C=1 :: CALL VCHAR(Y,X+2,32
,N*A)
1870 CALL D(Z,X,"%")
1880 CALL KEY(3,K,S):: IF S<
1 THEN 1880 ELSE IF K=13 THE
N CALL T :: CALL D(Z,X,"'"):
: SUBEXIT
1890 IF K=68 OR K=83 THEN CA
LL D(Z,X," "):: IF K=68 THEN
 C=C+1+N*(C=N)ELSE C=C-1-N*(
C=1)
1900 Z=Y+A*(C-1):: GOTO 1870
1910 SUBEND
1920 SUB WKEY(Y,X):: CALL D(
Y,X,"'taste'")
1930 CALL KEY(0,K,S):: IF S<
1 THEN 1930
1940 SUBEND
1950 SUB D(Y,X,A$):: DISPLAY
 AT(Y,X)SIZE(LEN(A$)):A$ ::
SUBEND
1960 SUB M(Y,A$):: CALL D(Y,
15-LEN(A$)/2,A$):: SUBEND
1970 SUB I(A$,C,P):: A$=SEG$
(A$,1,P-1)&CHR$(INT(C))&SEG$
(A$,P+1,9):: SUBEND
1980 SUB T :: CALL SOUND(9,9
99,4):: SUBEND
1990 SUB YN(A$,B$,R):: CALL
WIN(""," ",4,8,11,22):: CALL
 D(6,10,A$):: CALL D(8,10,B$
):: CALL CHO(6,9,2,R,10,10):
: SUBEND
2000 SUB ER(A$,B$):: CALL WI
N("fehler"," ",9,3,16,26)::
CALL M(11,A$):: CALL M(13,B$
):: CALL WKEY(15,12):: SUBEN
D
```

TI REVUE

Das Magazin für TI 99-4A

Nr. 11

Die DSR-Programmierung

Adventure Lösung

Test: Rave-Tastatur

Fill-Routinen für Ape-Soft Grafik

RGB-Encoder

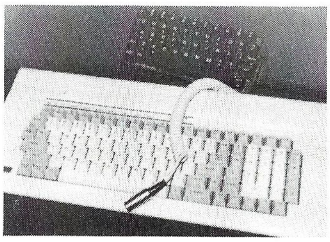
Tips & Tricks

Inhalt:

TI 99-4A:

TEST & TECHNIK

Tastatur de luxe
Tastatur-Set im Test Seite 33



Editor/Assembler
Neu bearbeitete Version des Modules nun auf Diskette erhältlich Seite 85

Miller's Graphics
Ein Name verpflichtet Seite 45

SERIE & SERVICE

Dem TI unter's Hemd geschaut
Wissenswertes über die TI 99/4A-Konsole Seite 37

RGB Encoder für den TI
Anschluß eines RGB Monitors an den TI: Schaltplan und Bauanleitung Seite 39

Laden ohne Error
Eine nützliche Hilfe beim Retten zerstörter Kassetten-Dateien Seite 42

Lösungshilfen für The Count
So bekommen Sie das Adventure in den Griff Seite 53

Die Programmierung der DSR
Dritter Teil des Programmierkurses Seite 63

LISTINGS

Music/Sound Generator
Ein hilfreiches Programm zum Editieren von Melodien und Soundeffekten Seite 54

Der Sprachsynthesizer zählt
Ein Listing zur korrekten Sprachausgabe von Zahlen Seite 58

Titelbilder für Video-Filme
Generator für Vor- und Nachspann selbsterstellter Videos Seite 59

EINFACH HERAUS-TRENNEN

Wichtiger Hinweis an alle TI-Abonnenten!

Nachdem der Verlag sich Anfang 1987 schweren Herzens entschließen mußte, den Titel TI REVUE als eigenständiges Objekt einzustellen und als TI-Sonderteil in die Zeitschrift HCA zu integrieren, haben wir allen Abonnenten mitgeteilt, daß wir ihr Abo nur noch bis Heft 12 berechnen. Sie haben jetzt das elfte von diesen zwölf Heften in der Hand.
Das heißt: Sollten Sie kündigen wollen, müßten Sie es jetzt tun, Sie wissen ja: Vier Wochen Kündigungsfrist vor Abo-Ablauf!
Selbstverständlich wird der TI aus der HCA nicht verschwinden, das sei Ihnen mit allem Nachdruck versichert.
Im Gegenteil: Es ist uns gelungen, mittlerweile wieder eine ganze Anzahl von Experten zu gewinnen, deren erste Arbeiten Sie hier in diesem Heft sehen.

DIE TI REVUE IST NICHT TOT!

Damit nicht genug. Wie wir informiert wurden, stellt ein Titel, der sich bislang ebenfalls noch mit dem TI 99/4A beschäftigte, in wenigen Wochen sein Erscheinen ein.

Damit ist HCA die einzige Zeitschrift im deutschsprachigen Raum, die sich auch weiterhin noch für den TI einsetzen wird.

Das bitten wir, bei Ihrer Entscheidung zu beachten.
Sollten Sie also weiterhin unser Leser bleiben wollen, werden Sie diesen Entschluß sicher nicht bereuen.

Die TI-REVUE wird weiterleben! Wir suchen noch gute Autoren!

IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Geschäftsführer:
Werner E. Seibt
Redaktion: Alfons Bauer,
Thomas Dehn, Christoph Kögler, Hartmut König, Jürgen Puschmann, Uwe Schröder, Lukas Stock
Verantwortlich für den Inhalt:
Lukas Stock (Chefredakteur)
Hartmut König (stv. Chefred.)
Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d
Alle: Postfach 1161
8044 Lohhof.
Anfragen bitte nur schriftlich
Es gilt die Honorarliste des Verlages.

Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.

Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

©by CA Verlag, Heßstraße 90, 8000 München 40
SPS und Autoren

TI 99/4A

BRANDHEISSE KNÜLLERPREISE: (SOLANGE VORRAT REICHT)

Hardware

Platine TI-99/4A (z.Teil defekt)	2 St. 39,–
Tastatur TI-99/4A	39,–
RS 232–Karte (Atronic)	249,–
Disccontroller DSDD (Corcomp)	349,–
P-Code-Karte (gebraucht)	399,–
80-Zeichen-Karte (Mechatronic)	449,–
Diskettenlaufwerk DSDD mit Einbausatz für Peripheriebox	679,–
Externe 128 K-Erweiterung mit Centronicsschnittstelle	249,–
Ext. 128 K-Erweiterung mit Centronicsschnittstelle + Druckerkabel + Brotherdrucker HR 5	449,–
dto. + Epsondrucker LX 800	849,–
dto. + Epsondrucker IX 800	1049,–
dto. + Epsondrucker LQ 800	1299,–
Axiom Centronicsinterface extern	129,–
RGB-Modulator mit Scartstecker	179,–
RGB-Modulator + Farbmonitor 1084	849,–
Grünmonitor Thomson (35 Mhz, Ton) mit Monitorkabel für TI-99/4 A	249,–
MBX-Sprachsteuereinheit + Baseball + Sewermania + Meteor Belt	229,–
Grafiktablett Supersketch + Buck Rogers + Defender	129,–
Supersketch + Defender + Dig Dug + Donkey Kong + Statistik	149,–
WIPO Mouse 99 anschlußfertig mit dt. Handbuch + Diskette	149,–
Akustikkoppler Dataphon S 21 d + Kabel für RS 232 Karte	279,–
Modulexpander 3-fach	125,–
Staubschutzhaube für TI-99/4A	20,–
Fernbedienung (2 Joysticks)	39,–
Joystickinterface + 2 Joysticks Competition Pro (Microschalter)	105,–

Bücher

Anwender-Handbuch Turbo Pascal dt.	39,–
Editor/Assembler Handbuch dt.	98,–
TI-99/4A Intern dt.	38,–
TI-Basic & Extended Basic	48,–
Assembler-Kurs (99er-Journal)	59,–
Introduction to Assembly Language	20,–

Software

Extended Basic II Plus	199,–
Mini Memory + Assemblerbuch dt.	199,–
Editor/Assembler (32 K + Disc notw.)	129,–
TI-Logo II (32 K + Disc notwendig)	149,–
Diskfixermodul (Navarone)	99,–
Database management (Navarone)	69,–
Terminal Emulator II + Statistik	59,–
Datenverwaltung/Analyse + Statistik	69,–
TI-99/4 A Flyer (Flugsimulation)	49,–

Modulgruppe A

Attack, Munchman, Othello, Invaders, Chisholm Trail, Tombstone City, Defender, Dig Dug	je 19,–

Modulgruppe B

Bigfoot, Balckjack & Poker, Espial, Fathom, Indoor Soccer, Moon Mine, Treasure Island, Donkey Kong, Jungle Hunt, Moon Patrol, Statistik	je 29,–

Modulgruppe C

Adventuremodul, Buck Rogers, Congo Bongo, Burgertime, Microsurgeon, Moonsweeper, Parsec, Return to Pirates Isle, Tunnels of doom, Video Chess, Protector II, Shamus	je 39,–
Je 1 Modul aus Gruppe A und B	44,–
Je 1 Modul aus Gruppe A und C	54,–
Je 1 Modul aus Gruppe B und C	63,–
Je 1 Modul aus Gruppe A, B und C	81,–
Popeye, Pole Position	je 49,–
TI-Artist (Diskette)	79,–
TI-Artist + Companiumdisketten 1,2 und 3 + Display Master	229,–
Graf-X + 3 Compan.disketten	199,–
Turbo Pascal (Diskette) + Anwenderhandbuch deutsch	149,–
Window Manager für Turbo Pascal	99,–
C 99 Compiler (Diskette)	99,–

Komplettübersicht über weiteres Zubehör + Modulpakete in unserer Preisliste !!!!!

Commodore / Schneider

Commodore C 128 D	969,–
Amiga 500 + Farbmonitor 1084	1.649,–
Amiga 2000 + Farbmonitor 1084	2.899,–
Schneider CPC 6128 Grünmonitor	729,–
CPC 6128 Farbmonitor	1.149,–

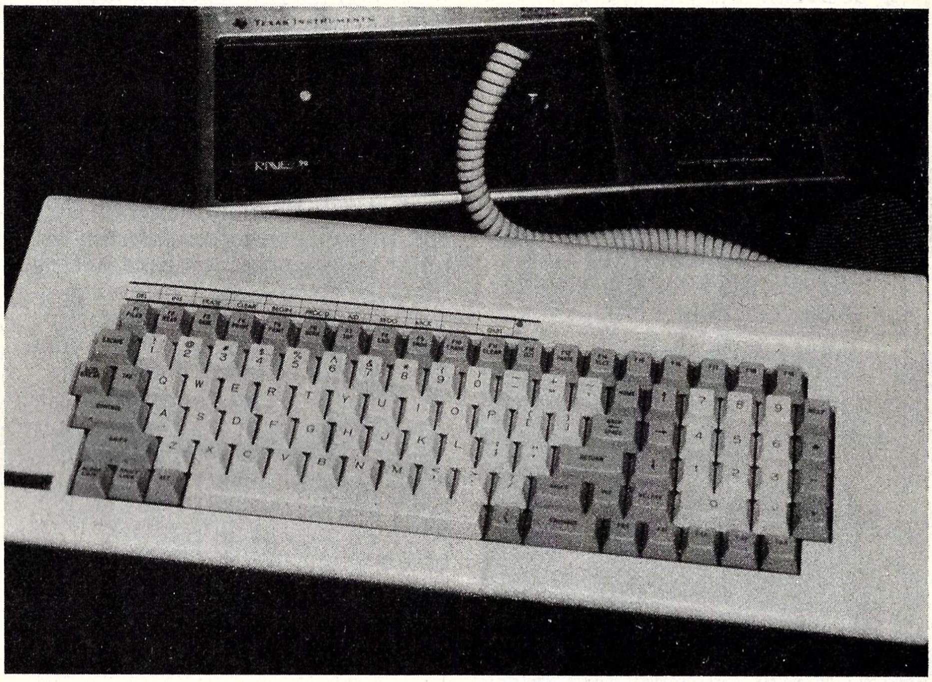
Versandkostenpauschale (Warenwert bis DM 1.000,– / darüber): Vorauskasse (DM 8,–/20,–), Nachnahme (DM 11,20/23,20), Ausland (DM 18,–/30,–). Lieferung nur gegen NN oder Vorauskasse; Ausland nur Vorauskasse. Preisliste (Computertyp angeben) gegen Zusendung eines Freiumschlags.

CSV RIEGERT

Schloßhofstr. 5, 7324 Rechberghausen, Tel. (07161) 52889

Tastatur de luxe

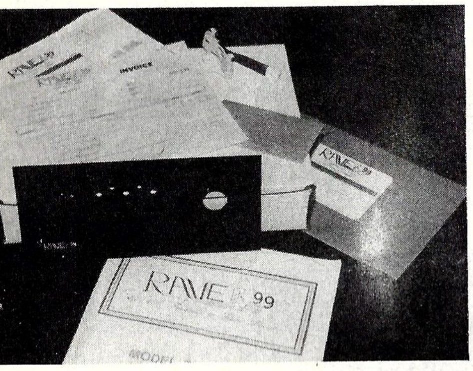
Die Tastatur des TI ist ja nun wirklich nicht als schlecht zu bezeichnen. Wer aber im Prinzip schon alle für ihn sinnvollen Erweiterungen hat, der sollte sich vielleicht mal Gedanken darüber machen, ob er sich nicht vielleicht mal eine Tastatur nach Art der PCs zulegen sollte.
Denn um genau so eine Tastatur handelt es sich bei dem Produkt, das die amerikanische Firma Rave 99 Co. vertreibt. Wer viel vor dem Computer sitzt, der wird sie nach einiger Zeit nicht mehr missen wollen.
Wenn man das Paket zum ersten Mal in die Hand bekommt, dann wundert man sich wohl, was denn an einer Tastatur dermaßen schwer sein kann. Beim Auspacken kriegt man aber sofort die Erklärung dafür: Die Tastatur ist nicht in einem Kunststoff-, sondern in einem stabilen Metall-Gehäuse untergebracht. Dazu kommt noch eine Platine, die an Stelle der TI-Tastatur eingebaut werden soll.
Aber immer hübsch eins nach dem anderen: Woraus besteht denn nun eigentlich das Tastatur-Set? Nun, wer das Komplett-Set bestellt hat, der erhält folgende Komponenten: Zuerst wäre da natürlich die Tastatur selbst. Sie ist, wie schon gesagt, in einem äußerst stabilen Gehäuse untergebracht und hat alles in allem 105 Tasten (man brachte die Typenbezeichnung 99/105!). An ihr ist ein Spiralkabel befestigt, das ca. 30 bis 40 cm lang ist (gestaucht). Des weiteren erhält man eine Platine, die die Tastendrücke auf der neuen Tastatur in die richtigen Informationen für den TI umsetzen soll. Diese Platine wird dort eingebaut, wo normalerweise die Tastatur des TI sitzt. Damit diese Platine nicht völlig offen in der Konsole zu sehen ist, wird dann auch noch eine Abdeck-Platte mitgeliefert, die diese Platine verdeckt und schützt.
Als letzten Baustein erhält man ein Kabel, das die Funktionen "Load Interrupt" und "Modul-Reset" ermöglicht. Dieses Kabel stellt übrigens die einzige Anforderung an die Lötkenntnisse des Users dar. Alles andere wird gesteckt oder geschraubt. Nicht vergessen bei dieser Auflistung sei allerdings noch die sehr gute Anleitung zum Einbau der Tastatur in den TI. Natürlich ist sie in Englisch gehalten, aber eine reiche Bebilderung macht das Lesen der Texte schon fast überflüssig. Zu guter Letzt sind da noch einige Kleinteile.



Die Rave-Tastatur im stabilen Metallgehäuse.

Der Einbau der Tastatur benötigt, wenn man sich Zeit dazu läßt, etwa 1/2 bis 3/4 Stunde. Dies allerdings auch nur, weil es einiges zu schrauben gibt. Der eigentliche Einbau ist eine Sache von Minuten. Ich will hier nun nicht die ganze Einbauanleitung wiedergeben, aber soviel sei gesagt: Zuerst muß die Unterseite der Konsole abgeschraubt werden, dann das Netzteil, die Hauptplatine und die Tastatur. Die alte TI-Tastatur muß abgezogen und an ihrer Stelle die Platine der neuen Tastatur eingebaut werden. Wenn diese Platine dann noch mit dem Netzteil und der Hauptplatine des TI verbunden worden ist, dann ist der Einbau im Prinzip schon vollbracht. Wer das Kabel für den "Load Interrupt" und den "Modul-Reset" einbaut, der kann noch ungefähr 5 Minuten addieren, bis auch das geschafft wäre. Wenn dann die ganze Konsole wieder zusammengeschraubt worden ist, kann man mit dem Austesten beginnen:
Wenn der TI nicht angefangen hat, Rauchringe auszustoßen, dann kann man endlich richtig loslegen. Wer sich wundert, daß die Tastatur anscheinend einige Tastendrücke nicht richtig interpretiert, der sollte überprüfen, ob er nach dem Einschalten der Konsole die Taste "ESCAPE" gedrückt hat. Diese Taste initialisiert das System. Im Normalfall kann man diese Taste jederzeit drücken, ohne daß etwas passiert, da "ESCAPE" nach dem Initialisieren der Taste "BACK" entspricht. (Man beachte: TI-Writer und Editor/Assembler.)
So, was für Tasten existieren denn nun eigentlich? Und muß man befürchten, daß es Schwierigkeiten mit bestehenden Programmen gibt? Nun, die zweite Frage läßt sich recht schnell mit „Nein" beantworten. Allerdings muß man dabei eine (unbedeutende) Einschränkung machen: Es gibt einige Assembler-Programme, die die Tastatur nicht über das "CALL KEY"-Pendant abfragen, sondern direkt über die CRU gehen. Auch benutzen einige den Modus 1 und 2 direkt nacheinander. Diese Programme kann man daran erkennen, daß mehrere Tasten auf einmal gedrückt werden können. Bei „Pinball" z.B. ist es ja dadurch möglich, mit "1" den linken und mit "=" den rechten Flipper hoch zu halten. Bei der Rave-Tastatur ist dies nicht mehr gleichzeitig machbar: Wenn die "1" gedrückt wird, gefolgt von "=", dann wird die "1" solange unterdrückt, bis "=" losgelassen wird. Dann erst kann man die "1" erneut drücken. Ein Loslassen von "=" aktiviert also auch nicht gleichzeitig die "1". Ähnliche Probleme tauchen auch bei dem Flugsimulator "SPAD XIII" auf. Aber diese Probleme sind nicht so ernsthaft, als daß sie als Hindernis für den Kauf dieser wirklich tollen Tastatur gehandelt werden sollten. »

An Tasten hat man wirklich mehr als reichlich. Natürlich sind alle Tasten der TI-Tastatur vorhanden, wenn auch in etwas anderer Anordnung; d.h., daß man ohne allzu großes Umdenken sofort mit dem Tippen anfangen kann. Die Vorzüge der Tastatur wird man dann auch nach und nach erkennen und nutzen. Wer sich noch nicht mit all den neuen Tasten anfreunden mag, der braucht sich eigentlich nur daran zu gewöhnen, daß die FCTN-Taste nicht mehr rechts der Leer-Taste, sondern links davon ist (da, wo vorher die CTRL-Taste lag, welche jetzt etwas weiter nach oben verschoben ist).



Was an Extra-Tasten als erstes ins Auge fällt, dürfte der sogenannte Zehner-Block sein. Auf ihm sind alle Ziffern und einige Rechenzeichen sowie der Dezimal-Punkt vorhanden. Dieser Block macht die Eingabe von Zahlen-Kolonnen (Pokelisten z.B.) extrem einfach. In einem anderen Testbericht war zu lesen, daß diesem Block die "Enter"-Taste fehle. Dies ist so nicht ganz richtig: Zwar liegt in der gewohnten rechten Ecke des Zehner-Blocks tatsächlich keine "Return"-Taste, dafür sind aber direkt links neben ihm sowohl der Abwärts-Pfeil (Fctn-E) wie auch die große "Return"-Taste plaziert. Sehr große Wege muß also niemand machen.
Der zweite Komplex, der auffällt, sind die sogenannten „Funktionstasten". Sie machen für den normalen Anwender die Taste "ALT" (besser bekannt unter "FCTN") eigentlich überflüssig. Mit ihnen lassen sich die Kombinationen "Fctn-1" bis "Fctn- =" allein durch Drücken einer Taste realisieren. Damit wären aber erst elf der insgesamt 24 Funktionstasten abgedeckt. Die restlichen 13 bekommen ihre Bedeutung erst, wenn man die Module „Multiplan" und/oder „TI-Writer" besitzt.
Beim TI-Writer z.B. erhalten die Tasten F11 bis F20 die Kombinationen Ctrl-1 bis Ctrl-0. Die restlichen Tasten erhalten Funktionen, die man zwar immer wieder braucht, auf der TI-Tastatur aber über reichlich unübersichtliche Kombinationen erreicht wurden. Wer sich darüber wundert, daß die Tasten unter den Modulen nicht unbedingt das machen, was sie eigentlich machen sollten, der sollte sich vergewissern, daß die Taste "SHIFT-LOCK" auch auf das jeweilige Modul eingestellt ist. Diese Taste ist übrigens nicht mit der "ALPHA-LOCK"-Taste zu verwechseln, die direkt daneben liegt. Apropos Alpha-Lock: Hierbei hatten die Erbauer einen ganz netten Einfall: Wenn diese Taste eingerastet ist, hat man (ganz normal) Großschrift eingestellt. Wenn man dann aber zusätzlich auch noch "SHIFT" mit einer anderen Taste drückt, bekommt man wieder Kleinschrift. "Shift" setzt also nicht nur hoch, sondern invertiert die Wirkung der Taste "ALPHA-LOCK".
Wer sich zu sehr an E-S-D-X in Verbindung mit der "FCTN"-Taste gewöhnt hat, der mag dabei bleiben. Sehr komfortabel auf der neuen Tastatur sind aber die Cursor-Tasten, die ebenfalls die "ALT"-Taste überflüssig machen. Die Aufteilung dieser Tasten braucht vielleicht etwas Übung, bis man sie eindeutig trifft, die Anordnung ist aber die gleiche wie bei E-S-D-X; nur, daß die Tasten unterschiedlich groß sind. Nett auch der Einfall, direkt unter den Cursor-Tasten noch einmal die Tasten "INS" und "DELETE" anzubringen. Dies dürfte ganz besonders die interessieren, die BASIC-Programme editieren wollen. Allerdings ist dies nur die Anwendung, die mir als erstes eingefallen ist. Für Full-Screen-Editoren (z.B. E/A) ist diese Anordnung ebenfalls äußerst hilfreich (als Assembler-Programmierer spreche ich dabei aus Erfahrung).
Soweit zu den eigentlichen Sondertasten. Genau genommen gibt es derer aber noch viel mehr. So z.B. die Tasten "TAB", "HOME" und "SCRL", die besonders im TI-Writer das Tippen noch einfacher machen, als es ohnehin schon mit der neuen Tastatur ist. Weiter gibt es auch noch "ENHANCE", welche den Charakter 255 liefert. Es ist mir zwar noch keine Anwendung dafür eingefallen, aber vielleicht kann man sie wirklich einmal gebrauchen. Dazu sei angemerkt, daß 255 auch gleichzeitig −1 bedeutet. Das Drükken dieser Taste erzeugt also einen Code, als ob keine Taste gedrückt worden sei. Wenn man aber im BASIC ist, bewegt sich der Cursor trotzdem weiter. Im Handbuch ist beschrieben (mit Beispiel), wie diese Taste benutzt werden kann.
Rein äußerlich fallen die Funktionstasten dadurch auf, daß sie im Gegensatz zu den „normalen" Tasten grau anstatt weiß sind. Das sieht nicht nur gut aus, sondern erleichtert auch die Suche. Schön wäre es ja gewesen, wenn über den Tasten F1 bis F11 noch eine Halterung für den Streifen mit ihren Bedeutungen angebracht worden wäre. Ich persönlich habe das jedenfalls gemacht: mit Doppel-Klebeband!
Alles in allem erhält man eine Tastatur, die eigentlich mehr ist als nur eine Tastatur. Wer viel mit dem TI-Writer oder dem Multiplan arbeitet, der sollte einmal durchrechnen, ob sich diese Anschaffung nicht für ihn lohnt. Aber auch jedem anderen TI-User sei diese Tastatur wärmstens empfohlen. Schön wäre es noch gewesen, hätten die Erbauer ein kleines RAM mit eingebaut, das als Buffer für den Tastendruck genutzt werden kann. Aber auch ohne diese Spielerei ist ihnen ein großer Wurf gelungen. Das einzig Störende an der ganzen Sache ist eigentlich nur der Preis: Ganze 262,75

NICHT GERADE BILLIG

US-Dollar will man für das Komplett-Set haben. Darin sind allerdings schon fast 40 Dollar allein für die Fracht enthalten. Da im Moment der Dollar freilich nicht übermäßig teuer ist, sollte man sich schnell überlegen, ob man nicht zugreifen sollte, denn es ist nicht anzunehmen, daß der Kurs weiter fallen wird.
Es soll übrigens möglich sein, sich auch nur die Platine mit der Abdekkung zu bestellen. In diesem Fall könnte man sich selber eine passende Tastatur hier in Deutschland kaufen. Im Prinzip dürfte jede normale PC-Tastatur angeschlossen werden können. Um die Platine gesondert zu kaufen, dürften ca. 150 US-Dollar ausreichen.
Wer an weiteren Informationen über die Tastatur interessiert ist, der kann sich an folgende Adresse wenden:
Michael Rittweger
Nissenstraße 12
D-2300 Kiel 14
Tel.: 0431/725566
Sie können sich auch direkt an die Vertriebsfirma wenden:
RAVE 99 Co.
112 Rambling Road
Vernon, CT 06066
USA
Ich würde mich freuen, wenn Sie mir über Ihre Erfahrungen mit dieser Tastatur berichten würden.

Michael Rittweger

Editor Assembler

Endlich ist sie da, eine wohl von vielen TI-Usern erwartete, überarbeitete und erweiterte Version des Editor-Assembler-Moduls. Ausgeliefert auf Diskette bietet sie für nahezu jede Systemkonfiguration eine Möglichkeit der Installierung. Die Umsetzung des Codes zur Erstellung eines Eprom-Moduls ist ebenfalls auf dieser Diskette.

Die neu überarbeitete und erweiterte Version des Editor/Assembler Moduls für den TI99/4A besteht aus einer Diskette. Da dieses E/A-Programm ebenso wie das Original von Texas Instrument in der Programmiersprache GPL geschrieben ist, setzt es entweder eine der inzwischen weit verbreiteten GRAM-Karten oder aber die in der TI-REVUE Nr. 5/85 vorgestellte Modulplatine voraus. Ein Diskettenlaufwerk sowie die 32KByte-Erweiterung werden, wie beim TI-Modul, ebenfalls benötigt. Bei den GRAM-Modulen wird das Programm bei Bedarf von Diskette geladen, bei der Modulplatine dagegen muß ein Eprom angefertigt werden.
Da man das Original TI-Modul als bekannt voraussetzen kann, sollen hier nur die Unterschiede und auch die neu im Basic enthaltenen 'CALL'-Befehle beschrieben werden. Ich habe diese Version längere Zeit getestet und keine Fehler oder Probleme (außer mit Turbo-Pascal) gefunden.
Sollte jemand dennoch Fehler feststellen, Vorschläge haben oder wissen, wie Inkompatibilitäten zu beheben sind, setze er sich bitte mit mir in Verbindung.

PSEUDO-RAMDISK

Dieses Programm ist Freeware, d.h., die Weitergabe an andere TI-User im Rahmen des privaten Programm-Austauschs ist erwünscht . . .
Wer die kleinen Vorzüge dieser Version zu schätzen weiß und ermessen kann, wie viel Arbeit in diesem Programm steckt, der sollte sich nicht zu schade sein, den Autor durch eine freiwillige kleine Spende in seiner Arbeit zu bestätigen.
Auf Anfrage (bitte die gewünschte Ladesoftware angeben) stehen in Kürze neue Erweiterungen zur Verfügung:

- mit der Möglichkeit, Editor und Assembler aus der GRAM zu kopieren, statt von Diskette laden zu müssen,
- mit den vom Minimem her bekannten Pseudo-RAM-Disks (jeweils nur ein File speicherbar),
- eine Sonderversion dieses Moduls speziell für den C-Compiler Clint Pulleys, die mit 48K Umfang praktisch alle wichtigen Files enthält (Laden von Disk entfällt!), ist bereits fertig . . .

Außerdem kann ein kompletter, assemblierbarer und kommentierter Quellcode des Mini Memory Moduls bei mir angefordert werden.

GPL AUF DER SPUR

Diese Diskette enthält das Modul in sechs verschiedenen Versionen, um möglichst jedem ein Laden mit seiner Ladesoftware für GRAM oder Eprom-Brenner zu ermöglichen.
Nachstehend eine kurze Beschreibung der sieben Files:

- "E/A":
  Programmfile, 8K direkt ausgelesen, ohne jede Änderung.
- "E/A_M":
  Programmfile mit 6-Byte-Header für die Ladesoftware der Mechatronic GRAM-Card in der Peripheriebox.
- "E/A_R":
  Programmfile mit Sechs-Byte-Header für die Ladesoftware des GRAM-Moduls vom TI99'er Workshop Rheinland.
- "FILE#EA04":
  Programmfile mit Vier-Byte-Header für die Ladesoftware von Hagenbuchner, sog. "I/O-Operations" bzw. "HEX2732".
- "OBJECT–":
  Uncompressed Objectcode zum direkten Laden in die GRAM.
- "OFFSET+":
  Compressed Objectcode mit Load-Offset, kann „normal" in die High Memory ab >C000 geladen und von dort mit geeigneten Programmen (z.B. Debugger) ins GRAM >6000 verschoben oder ins Eprom gebrannt werden.
- –"BSCSUP":
  Enthält eine auf knapp 2/3 Länge gekürzte, schnellere Version der bekannten „Basic Support Routinen" von der Assembler-Diskette: relocatibler compressed Objectcode.

ZUSÄTZLICHE "CALL"-BEFEHLE IM TI-BASIC

INIT LINK LOAD PEEK POKEV PEEKV CHARPAT
Diese Befehle wurden aus dem „Original"-Modul übernommen, nur sind bei CALL CHARPAT (C,C$,...) nun auch die ASCII-Codes 30, 31 gestattet.

GPOKE GPEEK
Diese beiden Befehle für GRAM/GROM-Speicher sind völlig analog aufgebaut wie POKEV/PEEKV für das VDP-RAM und werden ebenso aufgerufen.

NEW BYE
Diese beiden Befehle haben gleiche Funktion wie die TI-Basic-Statements gleichen Namens, nur können sie auch im Programm stehen. Da beide Befehle absolute Adressen in den Basic-GROMs #1, #2 benutzen, ist es denkbar, daß sie nicht mit jeder Konsolen-Version laufen.

DISPLAY (Zeile, Spalte, Ausdruck)
Erlaubt die Ausgabe eines Strings bzw. einer Zahl am Bildschirm, ähnlich dem Extended Basic-Befehl, nur ohne VALIDATE, SIZE, BEEP.
Die Bildschirmzeile wird nicht automatisch bis zu ihrem Ende gelöscht.
Spalte = 1 bis 28 (wie PRINT).

ACCEPT (Zeile, Spalte, Size, Variable)
Erlaubt den Input jeweils eines Strings oder einer Zahl am Bildschirm. Size muß immer angegeben werden, andere Optionen sind nicht möglich. Size kann positiv oder negativ sein (wie Ext.Basic). Bei Input einer Zahl werden ihr eventuell nachfolgende Buchstaben *ohne* Ausgabe einer Error-Meldung ignoriert.
Spalte = 1 bis 28 (wie PRINT).

EPSNPAT (Zeile, Spalte, String-variable)

Erlaubt die Ausgabe einer acht Byte langen Zeichen-Definition für Hardcopy auf Epson-kompatiblen Nadeldruckern im 8-Bit-Grafik-Modus. Die Steuerzeichen " ESC K <8> <0> " sind im Definitions-String nicht enthalten, um ein Anpassen auf andere Drucker oder ein Verketten zu längeren Strings zu erleichtern.
Spalte = 1 bis 32 (wie GCHAR).

COLSET (Farbe1, Farbe2, Farbe3)
Erlaubt das gleichzeitige Setzen von Vordergrund(1), Hintergrund(2) und Bildschirm/Rahmenfarbe(3) für die Charaktersätze #0 bis #12.

RESTORE (Zeile)
Erlaubt die Verwendung von numerischen Variablen zur Angabe der Zeilen-Nummer bei "RESTORE". Falls die genannte Zeile keine DATA-Zeile ist oder nicht gefunden wird, erfolgt Ausgabe einer Error-Meldung.

WAIT (Dauer)
Unterbricht das Programm für maximal die gewünschte Dauer oder bis zum Druck einer neuen Taste. Bei negativer Dauer ertönt ein "Beep" vor Beginn der Warteschleife. Die Dauer der jeweiligen Warteschleifen stimmt mit dem gleichnamigen Befehl des Mechatronic Extended Basic II überein.
Dauer = –16383 bis +16383

DEZ (HexString, num. Variable,...)
Erlaubt die Umrechung von mehreren, ein bis vier Zeichen langen Hexadezimalzahlen in Dezimalzahlen, sofern zuvor INIT ausgeführt wurde, sonst quittiert der Rechner das Fehlen der CIF-Routine mit "Can't do that".

HEX$ (Zahl, Stringvariable, ...)
Erlaubt die Umrechnung mehrerer Integer-Dezimalzahlen (–32767) bis +32767) in die entsprechenden Hexadezimal-Strings von immer vier Zeichen Länge. (Auch ohne CALL INIT funktionsfähig.)

BSCSUP
Lädt die „Basic-Support-Routinen" in die Low-Memory bis >28C6

("AORG"). Das Laden des gleichnamigen Objectcodes von Disk kann also entfallen. CALL BSCSUP muß *sofort* nach dem CALL INIT erfolgen, da die REFerences von zuvor geladenen Objectcodes nicht rückwirkend erkannt werden! Die Routine zeigt die Low-Memory-Belegung korrekt im "UTLTAB" an. Fehlendes CALL INIT, mehrfacher Aufruf oder laut "UTLTAB" fehlender LowMemory Speicherplatz führen zur Fehlermeldung "Can't Load".

CHARSET
Lädt alle Charakter von ASCII-Code 32 bis 127. Deutscher Zeichensatz mit Umlauten, echten Kleinbuchstaben.

CLRS
Löscht nur Bildschirmfenster von Spalte 2 bis 30 und setzt Edges, entspricht also: "CALL CLEAR :: CALL VCHAR(1,31,31,96) "

ÄNDERUNGEN IN HAUPTPROGRAMMEN:

EDITOR
Beim Verlassen des Editors durch <Back> oder <Quit> wird ein im Speicher befindliches Listing *nicht* gelöscht.
Der Editor kann unter der Option "4.List" nur noch Peripherie mit 80 Zeichen je Zeile ansprechen, nicht mehr den TI-Thermodrucker.
Bei der Option "4.List" erfolgt nun der Output im Dis/Var oder im Dis/Fix Format, je nach Art des Input Files. Die Routine arbeitet nun wesentlich schneller.
Der Editor arbeitet mit einem echten Kleinbuchstabensatz (ohne Umlaute) und den Farben Schwarz auf Grün.

ASSEMBLER
Beim Verlassen des Assemblers durch <Back> oder <Quit> wird dieser nicht im Speicher gelöscht. Ansonsten wurden nur unwesentliche Änderungen vorgenommen.

LOADER
Der Loader führt bei jeder Anwahl von "5. Run Pgm File" automatisch ein CALL INIT aus – und zwar erst nach der Eingabe eines Filenamens. Wird von einem Ladeversuch mit <Back> abgebrochen, bleiben alle Programme erhalten.
Der Loader führt *nicht* bei jeder erneuten Anwahl von "3. Load" ein CALL INIT aus, sondern nur, falls noch nie CALL INIT verwendet wurde. So bleiben alle geladenen Programme stets (auch nach "Quit") erhalten.
Die Option "4. Initialize" führt ein CALL INIT aus. Gleichzeitig wird der gesamte Speicher gelöscht (Highmem und Lowmem)! Lediglich der RAM/ROM-Bereich von >6000 bis >7FFF wird nicht gelöscht.

DISK-CATALOG
Der Disk-Catalog belegt keinen Speicher außerhalb des Scratch-PAD-RAM. Daher kann er keine geladenen Programme beeinträchtigen, d.h., Edit1, Assm1 oder geladene Files bzw. Editor-Listings bleiben erhalten.
Der Disk-Catalog wird durch Eingabe der Disk-Drive-Nummer gestartet und hat keine weiteren Funktionen (Rename, Protect, o.ä.).

ALLGEMEINES
Die bei "CALL INIT" geladenen Systemroutinen wurden verändert, um Speicherplatz zu sparen und möglichst oft "Jump" anstatt "Branch" benutzen zu können; auch wurden die so häufig benutzten Routinen VSBW und VSBR beschleunigt. Die Adressen aller Workspace-Register, Buffer und Vektoren wurden dabei unverändert gelassen, um maximale Kompatibilität zu erreichen.
Ebenso wurde eine schnellere, kürzere Version der "BSCSUP"-Routinen implementiert (siehe Basic-Info).
EDITOR und ASSEMBLER müssen nicht mehr von DSK1 geladen werden. Es werden "1" bis "5" als gültige Disk-Laufwerksnummern akzeptiert, um ein Betreiben mit (Horizon)-RAM-Disks oder Fremd-DiskControllern zu erleichtern.
Bei den Eingaben der File/Programm-Namen ist die "Alpha Lock"-Taste ohne Belang (Tastaturmode #3).
Alle Vorgaben sind selbstverständlich mit den Funktionstasten editierbar, auch wenn der Cursor meist nicht am Eingabe-Beginn erscheint. Außer <Clear> und <Begin> sind jetzt alle Funktionstasten belegt.

KOMPATIBILITÄT
Diese Version ist *nicht* kompatibel zum Turbo-Pascal '99.
Dagegen sind mit Clint Pulley's C99-Compiler sowie den meisten übrigen mir bekannten Programmen bisher keine Probleme aufgetreten.
Probleme können nur auftreten, bei:
- Benutzung absoluter Adressen im Modul-GROM/GRAM.
- Benutzung von Lowmem-Adressen ("EQU") anstatt "DATA"-Befehlen.
- Inkompatibilität der Programme mit dem Kleinbuchstabensatz.
- Überlappung von "AORG"-Codes mit den Adressen der implementierten "BSCSUP"-Routinen.

Probleme können nicht auftreten, bei:
- Verwendung absoluter Adressen für die "BLWP"-Routinen oder Buffer in der Low Memory.
- Verwendung des "BSCSUP"-Files von Diskette.

Winfried Winkler

Dem TI-99/4A unters Hemd geschaut

Die im Text verwendeten Zwischentitel Anatomie, Evolution und Revolution sind wörtlich zu nehmen. Revolutionäre Möglichkeiten bietet Ihnen Ihre TI-Konsole, wenn Sie die unten beschriebene Evolution durchgemacht hat.

Noch immer erfreut sich der TI 99-4A dank seiner guten Softwareorganisation großer Beliebtheit, und die Tatsache, daß noch eine große Anzahl von TI-Konsolen im Umlauf ist, die nur auf den Anschluß passender Peripherie wartet, die jedoch schwer beschaffbar ist, macht dieses Eigenbau- und -Umbaukonzept besonders interessant.
In diesem Bericht erfahren Sie näheres über
ANATOMIE:
Gründliche Analyse der TI-Konsolen-Hardware.
EVOLUTION:
Hardwareerweiterung für die TI-Konsole mit den Bestandteilen

- 16 KWort Speichererweiterung (16-Bit-breit)
- Vollständige CRU-Ausdekodierung
- Änderungen der Steuerelektronik in Vorbereitung auf den Abschnitt Revolution.

REVOLUTION:
Es wird eine Schnittstelle zu einer selbstgebauten Peripherie-Box beschrieben, die durch den Einsatz entsprechender Baugruppen völlig neue Gebrauchswerteigenschaften schafft:

- Errichtung eines 256-Byte breiten universellen parallelen I/O-Adreßraumes
- Errichtung eines zweiten 256-Byte breiten speziellen parallelen I/O-Adreßraumes zum Anschluß Z80-spezifischer Peripheriekarten
- Errichtung eines zweiten, unabhängigen Interruptsystems mit vektorisiertem Interrupt auf der Basis von Z-80-Peripheriebausteinen
- Möglichkeit der Beschleunigung von Byte-Zugriffen auf das Dreifache
- Baugruppe für GROM-GRAM mit dreifacher Zugriffsgeschwindigkeit –> Basic und Extended Basic mit Supergeschwindigkeit!
- Aufbau des von TI schon softwareseitig vorbereiteten G-Adreßraumes von 1 MByte!

1. ANATOMIE DER TI 99-4A-KONSOLE

1.0 Vereinbarungen

Es wird konsequent von der TI-üblichen Bezeichnungsweise Bit 0 = höchstwertiges Bit abgegangen und statt dessen die allgemein übliche Bezeichnungsweise Bit 0 = niederwertiges Bit angewendet, also:

höchstwertiges Bit –> niederwertiges Bit

A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

Die intern in der Konsole verwendeten Adreßleitungen werden mit Ax bezeichnet, die nach außen getriebenen Adreßleitungen mit ABx, die intern in der Konsole verwendeten Datenleitungen werden mit IDx bezeichnet, die nach außen betriebenen Datenleitungen mit EDx.

Alle Bezeichnungen werden in einem Schaltplan, „TI 99/4A-Hardwarekonzept“ verwendet, den wir in der nächsten Folge veröffentlichen. Sie sind ähnlich den Bezeichnungen der TI-Dokumentationen.

1.1 Taktgenerator

Der TMS 9900-Mikroprozessor benötigt einen 4-Phasentakt, der mit dem Schaltkreis U 601 – 74LS362 dem Mikroprozessor mit Zwölf-Voltpegel über die Anpaßwiderstände von 22 Ohm bereitgestellt wird. Der Quarzgenerator arbeitet mit einer Grundfrequenz von zwölf MH. Weiterhin gibt der Schaltkreis die vier Takte invertiert als /CLK1 ... /CLK4 mit TTL-Pegel aus, die zur Synchronisation der Konsolen-Hardware herangezogen werden. Nach außen wird am I/O-Connector nur /CLK3 bereitgestellt. Außerdem übernimmt der 74LS362 die Erzeugung des Power-On-RESET, das Signal wird vor seiner externen Verwendung über die Negatoren 508.2 und 508.2 getrieben und am Systembus bereitgestellt.

1.2 Der Videoprozessor

Die Konsole enthält zur Organisation des Bildaufbaus einen eigenen Videoprozessor: den U 100 – TMS 9929, der in Europa eingesetzt wird. Dieser Videoprozessor unterscheidet sich von dem in den USA verwendeten TMS 9918 durch die Ausgabe der Videosignale Y, R-Y und B-Y statt des Composite-Videosignals; dafür entfallen die beim TMS 9918 mögliche Ausgabe von CPU/CLK (Quarzfrequenz dividiert durch drei) sowie die Synchronisation mit einem externen Videosignal. Die folgende Tabelle zeigt die Unterschiede auf:

Pin	TMS 9918 (A)		TMS 9928/29 (A)	
35	B-Y	(Ausgang)	EXT-VDP	(Eingang)
36	Y	(Ausgang)	COMDID	(Ausgang)
38	R-Y	(Ausgang)	CPU-CLK	(Ausgang)

Der Videoprozessor arbeitet vollkommen unabhängig von der ihn umgebenden Hardware des eigentlichen Computers und verfügt über einen 16 KByte großen eigenen dynamischen Speicher, der u.a. auch als Arbeitsspeicher für den Computer mitgenutzt wird. Dieser Speicherraum ist jedoch nur als Datenspeicher erreichbar, dort können keine Maschinenprogramme laufen. Der Videoprozessor enthält die gesamte notwendige Hardware zur Ansteuerung des dynamischen RAMs. Über die Leitungen AD0 bis AD6 werden die Adressen zeitmultiplex und über die Leitungen /RAS und /CAS synchronisiert ausgegeben. Die Daten werden im Falle eines Schreibzyklus ebenfalls über die Leitungen AD0 bis AD6 und über die Leitungen D7 ausgegeben, im Falle eines Lesezyklus werden sie über die Leitungen RD0 bis RD7 gelesen. Die Leitung R/W’ liefert den erforderlichen Schreibimpuls beim Schreiben. Die Ausgänge der RAMs sind in dieser Be-

triebsart immer aktiv, daher müssen DI und D0 der RAMs auch getrennt geführt werden.
Der Videoprozessor liefert weiterhin alle Signale, die zur Erzeugung eines farbigen Fernsehbildes benötigt werden. Aus den Signalen Y, R-Y und B-Y läßt sich ein RGB-Anschluß realisieren oder über den mitgelieferten Modulator ein PAL-Fernsehsignal im UHF-Kanal 36.
Zur Kommunikation mit dem Mikrorechner verfügt der Videoprozessor über einen acht Bit breiten Datenbus CD0 bis CD7 sowie über die Statusleitung MODE, /CSR und /CSW und zur Synchronisation über eine Interruptleitung. Die Leitung MODE ist mit der Adreßleitunge A1 des Mikroporzessors verbunden. Ist sie Low zum Zeitpunkt eines über /CSR oder /CSW synchronisierten Zugriffs, handelt es sich um einen Datenzugriff; ist sie jedoch High, so wird auf eines der internen Statusregister zugegriffen. Wie die entsprechenden Signale /VDPWRITE und /VDPREAD erzeugt werden, wird im Abschnitt „Speicherbelegung" näher erläutert.
Der Videoprozessor arbeitet mit einem eigenen Takt von 10,7 MHz und gibt über den Ausgang Pin 37/GROMCLK einen Hilfstakt für den Betrieb der GROMs von TI mit einer Frequenz von Quarz / 24 aus.
Von der Initialisierung und dem Umgang mit dem VDP wird in einem gesonderten Beitrag die Rede sein.

1.3 Speicherbelegung der TI 99/4A-Konsole

Der TI 99/4A verfügt grundsätzlich über zwei Arten von zugriffsfähigem Speicher:

- CPU-Speicher: Dort können sowohl Daten als auch Maschinenprogramme angelegt werden. Dieser Bereich ist 64 KByte groß, wegen der später näher erläuterten Hardware sind jedoch nur 56 KByte wirklich nutzbar.
  Im CPU-Speicherbereich sind diverse I/O-Geräte memory-mapped eingeordnet; so der Soundgenerator, der Videoprozessor, das Sprach-Synthese-Modul und nicht zuletzt der nur als Datenspeicher nutzbare „G"-Speicherbereich.
- GROM bzw. GRAM: Das sind bis zu 16 Ebenen zusätzlich adressierbaren Datenspeichers von jeweils 64 KByte Umfang.
  Die „G"-Ebene wird über Toradressen, die im CPU-Speicherbereich memory-mapped angelegt sind, erreicht. Der „G"-Speicherbereich verfügt über einen eigenen Adreßzähler, der von der CPU manipuliert werden kann, ansonsten aber nach jedem Zugriff auf den „G"-Speicher selbständig inkrementiert. Nähere Informationen zum Arbeitsprinzip des „G"-Speicherbereiches werden bei der Besprechung der Steckeinheit „GROM-GRAM" im Abschnitt 3 – Revolution – gegeben.

1.3.1 CPU-Speicherorganisation

Der zur Verfügung stehende 64 KByte-Speicherbereich wird von der IS U504 (74LS138) in acht Segmente mit jeweils acht KByte Umfang unterteilt. Das erste Segment (>0000 ...>1FFF) enthält den 16 Bit breit angelegten ROM-Bereich der Konsole. Das fünfte Segment, das von der IS U505 (74LS138) in weitere acht Bereiche unterteilt wird, enthält den 16 Bit breit angelegten Scratch-Pad RAM-Bereich von 256 Byte im Bereich >8300 ...>83FF, der jedoch nicht vollständig ausdekodiert ist, und den Bereich >8000 auf den Soundgenerator, den Videoprozessor, den Sprachsynthese-Modul sowie die „G"-Speicherebene. Alle anderen Segmente sind in der Konsole nicht belegt und werden erst durch Erweiterungen über das Modul-Port oder das I/O-Port belegt:

>0000 ... >1FFF	Interner ROM-Bereich, selektiert durch die Leitung /ROMEN
>2000 ... >3FFF	Memory-Expansion, selektiert über die Leitung /EXL
>4000 ... >5FFF	DSR-Routinen von über das I/O-Port angeschlossenen Peripheriegeräten, selektiert über die Leitung /MBE
>6000 ... >7FFF	Maschinenroutinen von über das Modulport angeschlossenen Modulen, selektiert über die Leitung /ROMG
>8000 ... >9FFF	Scratch-Pad RAM sowie memory-mapped angelegte Peripherie, selektiert über die Leitung /MBI
>A000 ... >BFFF	Memory-Expansion, selektiert
>C000 ... >DFFF	über die Leitungen /EXH1,
>E000 ... >FFFF	/EXH2 und /EXH3

Aufteilung des CPU-Speicherbereiches >8000 ... >9FFF

>8000 ... >83FF	Scratch-Pad-RAM (>8300 .. >83FF)
>8400 ... >87FF	Soundgenerator, nur die Adresse >8400 wird benutzt.
>8800 ... >8FFF	Touradressen zum Videoprozessor
>8800	Lesen VDP-Daten
>8802	Lesen VDP-Statusregister
>8C00	Schreiben VDP Daten
>8C02	Schreiben BDP-Statusregister alle anderen Adressen werden nicht genutzt
>9000 ... >97FF	Toradressen zum Sprachsynthese-Modul
>9000	Lesen SPEECH
>9400	Schreiben SPEECH alle anderen Adressen werden nicht genutzt
>9800 ... >9FFF	Toradressen zum „G"-Speicherbereich
>9800	Lesen G-Daten Ebene 0
>9802	Lesen G-PC
>9804	Lesen G-Daten Ebene 1
>9806	Lesen G-PC
>9808	Lesen G-Daten Ebene 2
>980A	Lesen G-PC
*	*
*	*
>983C	Lesen G-Daten Ebene 15
>983E	Lesen G-PC
>9C00	Schreiben G-Daten Ebene 0
>9C02	Schreiben G-PC
>9C04	Schreiben G-Daten Ebene 1
>9C06	Schreiben G-PC
>9C08	Schreiben G-Daten Ebene 2
>9C0A	Schreiben G-PC
*	*
*	*
>9C3C	Schreiben G-Daten Ebene 15
>9C3E	Schreiben G-PC

Die Unterscheidung der vom Betriebssystem unterstützten 16 Ebenen des „G"-Speicherbereiches wird von der Konsole nicht vorgenommen, zur Nutzung dieser Möglichkeiten sind nähere Informationen im Beitrag zur Steckeinheit „GROM-GRAM" im Abschnitt Revolution enthalten.

Heinrich Cronjaeger

RGB Encoder für den TI-99/4A

Einem Hochfrequenzsignal, wie dem eines Videoprozessors, schadet jede weitere Bearbeitung erheblich. Wenn es gelingt, die bei Rechner wie Monitor vorhandenen Modulatoren und ZF-Verstärker zu umgehen, strahlt die Bildschirmausgabe Ihres TI auf einem RGB-Monitor in nie gekannter Brillanz.

Wer hat sich nocht nicht darüber geärgert, daß der Video-Display-Prozessor TMS 9929A nur das Helligkeitssignal Y und die Farbdifferenzsignale R-Y und B-Y zur Verfügung stellt, so daß kein handelsüblicher Monitor verwendet werden kann? Die meisten Monitore arbeiten mit den drei Farbsignalen Rot, Grün und Blau, welche dann über Verstärker direkt an der Bildröhre anliegen. Die höchstmögliche Auflösung hängt nur noch von der Bandbreite der Verstärker und der Qualität der Bildröhre ab. Im Gegensatz zum normalen Fernsehapparat entfällt also der Signalweg über Modulator, Tuner und ZF-Verstärker, wodurch eine hervorragende Bildqualität erreicht wird. Um nun einen Monitor am TI anschließen zu können, wurde ein RGB-Encoder konzipiert. Das Herz dieses Encoders stellt der Ein-Chip-Videoprozessor TDA 3506 dar. Mit diesem Baustein und wenigen externen Bauelementen ist es möglich, aus den Helligkeits- und Differenzsignalen die not-

RGB-MONITOR AM TI

wendigen Ansteuerspannungen für einen Monitor zu erzeugen. Außerdem liefert der Encoder noch positive und negative einstellbaren Synchronimpulse.

DIE SCHALTUNG

Der TDA 3506 wird sehr oft in Fernsehgeräten eingesetzt. Für unsere Zwekke modifiziert, kann die externe Beschaltung noch um einige Bauteile minimiert werden, weil die Regelspannungen, die der Prozessor liefert, für eine Monitoranwendung uninteressant sind.

Für eine einwandfreie Funktion sind dem Videoprozessor folgende Eingangsspannungen zuzuführen:

1. Helligkeitssignal Y
2. Differenzsignal R-Y
3. Differenzsignal B-Y
4. Sandcastle-Impuls.

Außer dem Sandcastle-Impuls liegen alle Eingangsspannungen für den Encoder am Modulatoranschluß des Computers an (siehe Bild 1).

Dieser Sandcastle-Impuls ist ein mehrstufiger Impuls, der zur Austastung der horizontalen und vertikalen Synchronimpulse sowie des Bursts auf der hinteren Schwarzschulter des FBAS-Signales dient. Es gibt verschiedene Möglichkeiten, diesen Impuls zu gewinnen (z.B. TDA 2593). Die billigste davon ist die Erzeugung über monostabile Kippstufen, bidirektionale Schalter und Zenerdioden.

Über eine Impulsabtrennstufe, welche mit dem Trasistor BC 177 realisiert wurde, muß der Bildanteil des Y-Signals von den Synchronimpulsen getrennt werden. Die gewonnenen Impulse gelangen an die Eingänge der Monoflops von IC 2.

Am Pin 6 des IC 4066 liegt damit eine Schaltspannung an, die um den Horizontalimpuls verzögert und von der Dauer her einstellbar ist. Die Zenerdiode Z 6,8 wird im Takt dieses Pulses überbrückt. Am Pin 12 des IC 4066 kommt ein Impuls zustande, der mit der positiven Flanke des Horizontalimpulses getriggert wird und ebenfalls von der Dauer her einstellbar ist. Somit ist die Grundform des Sandcastle-Impulses hergestellt. Da aber in der vertikalen Austastlücke ein anderes Gleichspannungsniveau gefordert wird, ist noch eine Schaltspannung notwendig, die genau in dieser Austastlücke das Gesamtniveau auf ca. 2,5 V erhöht.

SO WIRD'S GEMACHT

Dies wird mit dem NAND-Gatter und IC 3 erreicht. Das Gatter erkennt den Vertikal-Austastimpuls und triggert damit das Monoflop von IC 3. Auch diese Schaltdauer kann mit einem Trimmer einjustiert werden, so daß die Zenerdiode Z 2,7 bei Bedarf, sprich V-Lücke, zugeschaltet wird. Diese kurzen Erläuterungen sind anhand eines Impuls-Zeitdiagrammes besser zu verstehen (siehe Bild 2).

Durch das Zu- und Abschalten der Zenerdioden stellt sich am 2,2 kOhm Widerstand, entsprechend den Impulsdauern, der erforderliche Spannungswert ein:

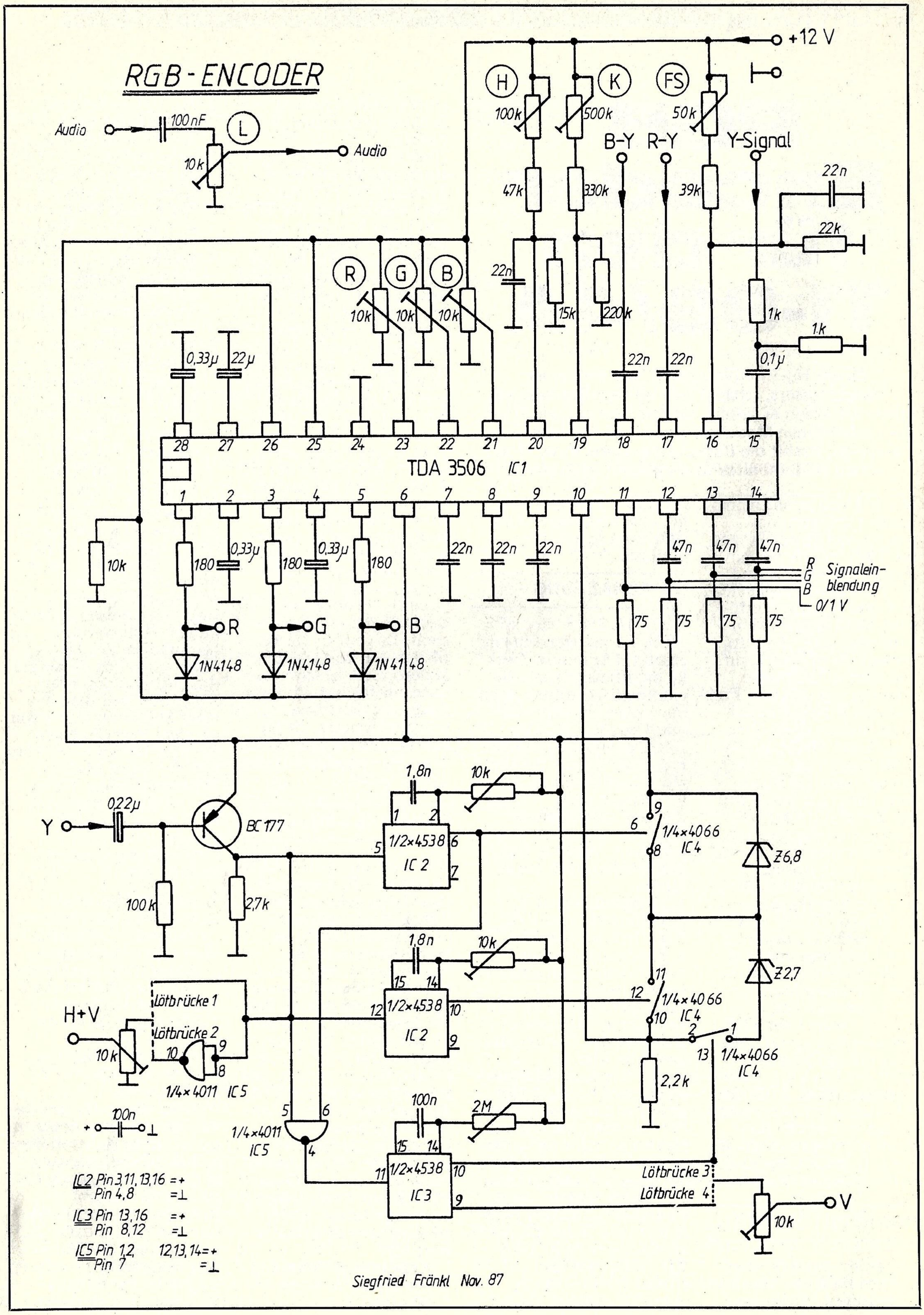
1. Kein Impuls: Während des Bildinhaltes, ergibt 0 V.
2. Pin 12 von IC 4066 aktiv: Über die Z 6,8 und dem elektronischen Schalter fallen ca. 7,5 V ab, so daß am Widerstand 4,5 V anliegen.
3. Pin 12 und Pin 6 von IC 4066 aktiv: Die Zenerdioden werden überbrückt und es tritt nur ein Spannungsabfall über die Schalter auf. Am Widerstand liegen ca. 11 V an.
4. Pin 13 von IC 4066 aktiv: Die vertikale Austastlücke wurde erkannt. Für ca. 1 ms fallen über die Zenerdioden, wenn die beiden Schalter offen sind, 9,5 V ab. Es stellt sich am Widerstand 2,5 V ein.

Wie dieser Sandcastle-Impuls sowohl mit als auch ohne Oszilloskop einzustellen ist, wird in der Abgleichanleitung am Schluß dieser Bauanleitung beschrieben. Liegen nun alle Impulse und Spannungen am TDA 3506 an, liefert dieser die drei Signalspannungen Rot, Grün und Blau mit einer Signalamplitude von ca. zwei Vss. Um einen Monitor betreiben zu können, fehlt uns noch die Synchronisation. Für einen möglichst universellen Einsatz des RGB-Encoders stehen mehrere verschiedene Synchronimpulse bereit:

1. positive Horizontal- und Vertikalsynchronimpulse = Lötbrücke 1
2. negative Horizontal- und Vertikalsynchronimpulse = Lötbrücke 2
3. positive Vertikalsynchronimpulse = Lötbrücke 3
4. negative Vertikalsynchronimpulse = Lötbrücke 4

Welcher davon für den jeweiligen Monitor notwendig ist, steht sicher in der Bedienungsanleitung oder in den Schaltungsunterlagen des Gerätes.

Die Auswahl eines dieser Impulse erfolgt dadurch, daß man auf der Platine den Trimmer T 11 oder T 12 mit der entsprechenden Zuführungslei-



RGB-ENCODER
Audio
100 nF
L
10 k
Audio
+12 V
H
K
FS
100k
500k
50k
B-Y
R-Y
Y-Signal
47k
330k
39k
22n
22k
22n
15k
220k
1k
1k
0,1µ
R
G
B
10k
10k
10k
0,33µ
22µ
22n
22n
TDA 3506
IC1
10k
180
0,33µ
180
0,33µ
180
22n
22n
22n
47n
47n
47n
R
G
B
Signaleinblendung
0/1 V
75
75
75
75
R
G
B
1N4148
1N4148
1N4148
1,8n
10k
0,22µ
Y
BC 177
1/2×4538
IC 2
1/4×4066
IC 4
Z 6,8
100 k
2,7k
1,8n
10k
Z 2,7
Lötbrücke 1
H+V
Lötbrücke 2
10 k
1/4×4011 IC5
1/2×4538
IC 2
1/4×4066
IC 4
1/4×4066
IC 4
2,2 k
100n
2M
100n
1/4×4011
IC5
1/2×4538
IC3
Lötbrücke 3
Lötbrücke 4
V
10k
IC2 Pin 3,11,13,16 = +
Pin 4,8 = ⊥
IC3 Pin 13,16 = +
Pin 8,12 = ⊥
IC5 Pin 1,2, 12,13,14 = +
Pin 7 = ⊥
Siegfried Fränkl Nov. 87

tung der Impulse durch eine Lötbrücke verbindet. Damit stehen uns nun alle Signale zur Verfügung, um jeden RGB-Monitor anschließen zu können.

WAS TUN, WENN KEIN MONITOR VORHANDEN IST?

Natürlich ist der Encoder auch für Fernsehgeräte geeignet, die mit RGB-Ansteuerung arbeiten. Hier bedarf es jedoch einiger fernsehtechnischer Kenntnisse, um den Encoder anpassen zu können.
Am Beispiel eines Grundig-Portable soll kurz erläutert werden, wie dies zu realisieren ist. Es handelt sich hier um ein Gerät, welches ebenfalls einen Videoprozessor beinhaltet. In diesem Fall ist es der TDA 3562, der genauso wie der TDA 3506 oder der TDA 3560 externe RGB-Einblendeingänge besitzt, die über eine 1-V-Schaltspannung aktivierbar sind. Diese Einblendeingänge sind normalerweise gleichspannungsmäßig mit 100 nF Kondensatoren entkoppelt. Außerdem befindet sich vor jedem Kondensator ein 75 Ohm-Widerstand gegen Masse. Nachdem die drei Widerstände entfernt wurden, kann man die RGB-Ausgänge des Encoders mit den Einblendeingängen o.g. Bausteine verbinden (zur Kontrolle: die gesamte Stromaufnahme des Encoders sollte ca. 100 bis 110 mA betragen).
Legt man eine Schaltspannung von 1 V an Pin 9 des TDA 3562 (Pin 9 TDA 3560, Pin 11 TDA 3506), so wird das vom Encoder kommende Signal durchgeschleift.

Bild 1:

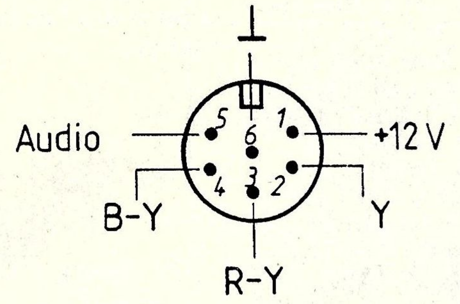
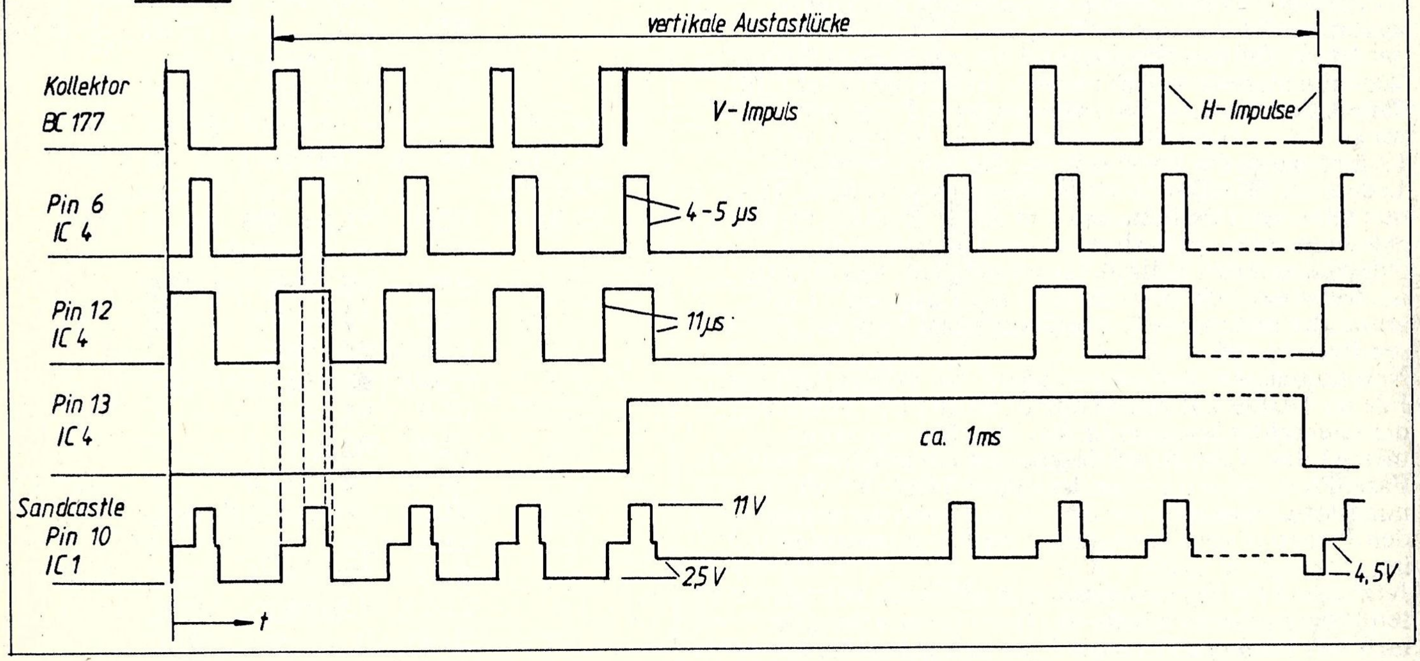


Bild 2:



Die Synchronisierung erfolgt dadurch, daß man die negativen H+V-Impulse (= Lötbrücke 2) bzw. das Y-Signal an den FBAS-Eingang des Ablenkbausteins TDA 2595 o.ä. legt.
Das Audiosignal des Encoders muß vor der Ton-Endstufe des Fernsehers eingespeist werden.
Als FM-ZF-Verstärker und Demodulator wird sehr oft der Baustein TBA 120 T eingesetzt. Dieser besitzt einen externen Toneingang an Pin 3. Wird hier das Audiosignal eingespeist und gleichzeitig Pin 2 des TBA 120 T auf Masse gezogen, so wird der Ton des TI99 an die Endstufe des Fernsehers durchgeschleift.
Mit einem Umschalter 3x2 ist es möglich, den Fernseher entweder als Monitor oder als normalen Empfänger zu betreiben.
Da sicher nicht jeder ein Fernsehgerät mit all diesen integrierten Schaltungen besitzt, sind die Erläuterungen hierzu sehr kurz gehalten. Sie sollen nur eine von vielen Möglichkeiten darstellen, ein Fernsehgerät als Monitor einzusetzen, denn auch damit sind enorme Verbesserungen bezüglich der Auflösung zu erreichen. Hier kann sich jeder Fernsehtechniker weiterhelfen.
In der nächsten Ausgabe der TI-REVUE wird der Aufbau der Schaltung sowie der Abgleich besprochen. Auch ein Platinenlayout zu dieser Schaltung und eine Bauteileliste wird vorgestellt, um allen TI99/4A-Usern diese Schaltung zugänglich zu machen. *Siegfried Fränkl*

Laden von Kassette ohne Error: Noerrload

Wenn man aus Versehen Teile eines Cassettenfiles zerstört hat, ist das immer ausgesprochen ärgerlich, da der TI nicht dazu zu bewegen ist, den Rest noch einzulesen. Aber auch, wenn ein File laufend an einer bestimmten Stelle einen "Error Detected in Data" liefert, obwohl es vom Hören her scheinbar in Ordnung ist, kann man schnell die Geduld verlieren. Deshalb hier nun die Abhilfe: Eine exakte Nachbildung der Laderoutine aus dem ROM mit nur zwei kleinen Änderungen. Erstens: Keinerlei Errorabfrage mehr! Zweitens: Abbruch des Ladens ist jederzeit durch die Taste SHIFT möglich!

Das Programm benötigt das Mini-Memory und den Cassettenrecorder ohne weitere Zusätze. Nach dem Laden der Routine aus dem EASY-BUG heraus schreiben Sie in >7114 die maximal mögliche Länge des zu ladenden Files und in >7116 die VDP-Startadresse, ab der es geladen werden soll (beispielsweise Länge >3000 und Startadresse >1000). Dann starten Sie das Programm LOAD (Adresse finden Sie in der REF/DEF-Table) mit Execute. Ohne Ausgabe von Meldungen wartet der TI nun auf Ihr File, von dem zumindest der letzte Teil des Vortons und die ersten zwei Datensätze noch erhalten sein müssen. Ist das nicht der Fall, dann laden Sie dies von einem anderen Programm etwa gleicher Länge und wechseln dann während des Ladevorgangs die Bänder. Files, die beim normalen Laden mit "No Data found" abgewiesen wurden, können selbstverständlich auch von dieser Routine nicht gelesen werden. Ein Abbruch der Ladefunktion ist jederzeit mit den SHIFT-Tasten möglich, was wichtig ist, wenn der Schluß eines Files fehlt und der TI aufgrund der fehlenden Errorabfrage sonst ewig auf diesen Rest warten würde.

Nun kommt das Zeitaufwendigste: Sie müssen sich das File im VDP-RAM ansehen und, wenn noch möglich, die Ladefehler korrigieren. Bei Maschinenprogrammen ist das in der Regel durchaus noch möglich, bei Basic-Programmen ist ein derartiger Versuch hoffnungslos und es hilft nur beten, daß nach der folgenden Prozedur noch etwas mit dem File anzufangen ist.

Jetzt saven Sie die geladenen und ggf. auch korrigierten Programmteile wieder ab. Dazu suchen Sie sich im VDP-RAM das Ende des Files und geben die daraus resultierende exakte Länge in >7114 an. Dann starten Sie die Routine SAVE (Adresse finden Sie ebenfalls in der REF/DEF-Table) mit Execute. Ohne Ausgabe von Meldungen saved der TI nun sofort das File neu ab.

Handelte es sich um ein Basic-File, dann kommt jetzt das große Bangen: Ist da noch etwas herauszuholen? Sie laden das neu gesavte File mit OLD CS1 und geben LIST "RS232" bzw. LIST "PIO" ein. Mit etwas Glück listet der TI nun das (auf keinen Fall mehr lauffähige!) Programm auf den Drucker aus und stürzt dabei in irgendeiner Zeile ab. Nach FCTN + geben Sie in einem zweiten Versuch dann LIST "RS232": Zeile– ein, wobei mit „Zeile" eine Zeilennummer größer als die zuletzt gedruckte gemeint ist.

So können Sie dann Stück für Stück ein BASIC-Programm wieder bekommen, das Sie anschließend neu eintippen müssen. Daran sehen Sie auch, was NOERRLOAD bezüglich BASIC-Programmen kann und was nicht: Es kann nur sinnvoll angewandt werden, wenn es sich um ein Programm handelt, von dem noch kein Listing existiert (weil es z.B. selbst erdacht ist) und auch leichtsinnigerweise keine Backup-Kopie. In allen anderen Fällen ist das erneute Eintippen des Listings in jedem Falle die bessere Alternative.

Sven Dyroff

```
       DEF  LOAD
       DEF  SAVE
       AORG >7118
BACK   BSS  2
*---------------------------
LOAD   MOV  R11,@BACK
       STWP R11
       MOV  R11,@QUIT+2
       LWPI >83E0
       CLR  R12
       SBZ  18
       SBZ  19
       SBZ  20
       LI   R1,>7114
       B    @X142E
*---------------------------
X1344  DATA >0010
*X1346
X13BA  MOV  *R1+,R5
       AI   R5,>003F
       SRL  R5,R6
       SOC  *R1,R0
       MOV  R0,R10
       MOVB @>83E1,*R15
       CLR  R1
       CLR  R12
       MOVB R0,*R15
       SOC  @>0032,R14
       SBZ  2
       SBZ  12
       LDCR R3,15
       SBZ  0
       SBZ  1
       SBO  3
       B    *R11
*X13E2
*X1404
X1424  DATA >2100
*X1426
X142E  SZC  @>1344,R14
       LI   R0,>4000
X1436  LI   R3,>002B
       BL   @X13BA
       MOV  R10,R7
       CLR  R0
X1442  MOVB @X1442+1,@>837C
X1448  LI   R8,>7530
       LIMI >0001
```

```
         LI   R6,X1458
X1454    LI   R3,>002B
X1458    ANDI R1,>00FF
         DEC  R8
*NOERR   JEQ  X155E          *ERROR weil NO DATA FOUND
         LI   R2,>0030
         MOV  R0,R0
         JNE  X146A
         A    R2,R2
X146A    BL   @X1572
         JMP  X1472
         JMP  X1458
X1472    DEC  R2
         JNE  X146A
         LI   R9,>7FFF
         LI   R8,>0008
         LDCR R9,15
         SBZ  0
         SBO  3
X1484    BL   @X15BA
         JMP  X148C
         JMP  X1484
X148C    DEC  R8
         JNE  X1484
         SBO  0
         STCR R3,15
         S    R3,R9
         MOV  R9,R3
         SLA  R9,2
         A    R9,R3
         SRL  R3,6
         ORI  R3,>0001
         LI   R10,X14B0
         CI   R3,>001F
         JLT  X1454
         B    @X1580
X14B0    BL   @X1572
         JMP  X14B0
         LI   R2,>0007
X14BA    BL   @X1572
         JMP  X1458
         DEC  R2
         JNE  X14BA
         LI   R6,X14F8
         MOV  R0,R0
         JNE  X152E
         MOVB @X1424,@>837C
         MOV  R7,R0
         CLR  R7
         BL   @X15A0
         C    R5,R4
*NOERR   JL   X155E          *ERROR weil NOT ENOUGH MEMORY
         MOV  R4,R5
         INC  R5
         NEG  R7
         BL   @X15A0
*NOERR   JNE  X155E          *ERROR
         JMP  X1526
```

LIST IN BASIC- UND ASSEMBLER-ROUTINEN?

Ich habe einige Fragen an Sie:
1. Wie kann der LIST-Befehl aus einem Basic-Programm heraus aktiviert werden?
2. Der LIST-Befehl aktiviert im TI irgendeine Mimik, die das im Speicher befindliche codierte BASIC-Programm decodiert und am Bildschirm sichtbar macht. Läßt sich diese Mimik aus Assembler heraus aktivieren?
3. Gibt es zu diesem Zweck eventuell Poke-Adressen?
4. Worin besteht der Unterschied zwischen Assembler und GPL?
5. Gibt es ein Programm für die Erstellung von GPL-Programmen, ähnlich dem Line-by-Line Assembler für Minimem?
6. Gibt es Anwendungen für GPL-Programme?

Helmut Fiedler, Baar

Auf Ihre vielfältigen Fragen gibt es der Reihe nach folgende Antworten:
1. Unter bestimmten Bedingungen können Sie durch Starten des GPL-Interpreters mit der Adresse G4018 den Befehl LIST aus einem Basic-Programm mittels Assembler-Unterprogramme aktivieren, was jedoch eine genaue Kenntnis der TI-GROMs voraussetzt, zumal sich auf keinen Fall verhindern läßt, daß der TI nach dem Listen das Programm beendet und in den Eingabemodus zurückkehrt.
2. Aktivieren läßt sich diese Mimik nicht, bestenfalls nachbilden, da in Assembler die entsprechenden Basic-Pointer fehlen. Ein Blick in Heiner Martins ROM-Listing von G474C bis G481D zeigt aber sofort, daß sich der Aufwand einer Neuprogrammierung kaum lohnt.
3. Nein (tut mir leid!).
4. a) Assembler ist die Sprache des Prozessors, nur die allein versteht er. GPL versteht er hingegen nicht, deshalb muß er einen Interpreter abarbeiten, der sie ihm verständlich macht.
b) GPL ist leistungsstärker und einfacher, aber langsamer als Assembler, da das Abarbeiten des Interpreters Zeit braucht.
c) Für den Rest siehe den GPL-Bericht, Teil VIII.
5. Ja: Den GPL-Assembler von Michael Weiand, der auch schon in der TI-REVUE in Anzeigen vorgestellt wurde. Benötigt werden dazu Ed/Ass, 32 K und mindestens ein Laufwerk.
6. Die Frage ist schnell beantwortet: Fast jedes TI-Modul ist zum größten Teil in GPL geschrieben, vom Schachmeister bis zum Ed/Ass und Ext.-Basic.
Deshalb Anwendungen en masse.

Sven Dyroff
TI-Workshop
Wiesbaden
i.A. TI-REVUE

```
X14EC  ANDI R7,>00FF
       NEG  R7
       BL   @X15A0
       JEQ  X1506
X14F8  MOV  R5,R5
*NOERR JLT  X155E         *ERROR
       MOVB @>83E1,*R15
       NEG  R5
       MOVB R0,*R15
       JMP  X1448
X1506  MOV  R5,R5
       JLT  X151A
       LI   R2,>0049
X150E  LI   R6,X1516
       BL   @X15A0
X1516  DEC  R2
       JNE  X150E
X151A  AI   R0,>0040
       MOVB @>83E1,*R15
       ABS  R5
       MOVB R0,*R15
X1526  CLR  R7
       DEC  R5
       JNE  X1448
       JMP  X1558
X152E  LI   R2,>0040
       CLR  R7
X1534  BL   @X15A0
       SWPB R4
       COC  @X1344,R14
       JNE  X154E
       SB   @>FBFE(R15),R4
       JEQ  X1552
       CI   R5,>0001
       JEQ  X1552
       JMP  X14F8
X154E  MOVB R4,@>FFFE(R15)
X1552  DEC  R2
       JNE  X1534
       JMP  X14EC
X1558  MOVB @X1436+2,@>837C
X155E  SZC  @X1344,R14
       SZC  @>0032,R14
       SBZ  3
       SBO  12
       SBO  1
       SBO  2
QUIT   LWPI >0000
       MOV  @BACK,R11
       B    *R11
X1572  MOV  R11,R10
X1574  JMP  X1574
       BL   @X15BA
       INCT R10
       ORI  R1,>FF00
X1580  CZC  @X1458+2,R1
       JEQ  X158C
X1586  TB   >001B
       JNE  X1590
       JMP  X1586
X158C  TB   >001B
       JNE  X158C
X1590  LDCR R3,15
       SBZ  0
       SBO  3
       ANDI R1,>00FF
       XOR  @X1458+2,R1
       B    *R10
X15A0  LI   R8,>0008
       CLR  R4
       MOV  R11,R9
X15A8  SLA  R4,1
       BL   @X1572
       JMP  X15B2
       INC  R4
X15B2  DEC  R8
       JNE  X15A8
       A    R4,R7
       B    *R9
X15BA  TB   8                *KSCAN
       JEQ  $+6
       B    @X1558          *END
       TB   >001B
       JEQ  X15CA
       CZC  @X1458+2,R1
       JEQ  X15D0
X15C4  XOR  @X1458+2,R1
       B    *R11
X15CA  CZC  @X1458+2,R1
       JEQ  X15C4
X15D0  INCT R11
       B    *R11
*------------------------
SAVE   MOV  R11,@BACK
       STWP R11
       MOV  R11,@QUIT+2
       LWPI >83E0
       BL   @>0864
       MOVB @>8373,R1
       SRL  R1,8
       AI   R1,>8302
       MOV  @ADR,*R1+
       MOV  @ADR+2,*R1
       SWPB R1
       MOVB R1,@>8373
       BL   @>0842
       LI   R1,>7114
       B    @>1346
ADR    DATA QUIT2-2
       DATA >003A
QUIT2  DECT @>8373
       BL   @>0842
       CLR  @>837C
       B    @QUIT
       END
```

Miller's Graphics

Um es gleich vorweg zu nehmen: Ursprünglich sollte dieser Artikel die Besprechung eines Programms aus dem Miller's Graphics Triumvirat Advanced Diagnostics, Explorer und Disk-Assembler sein, jedes für sich ein Juwel auf dem TI-Softwaremarkt. Doch die Besprechung eines MG-Programms muß diesem gerecht werden und entsprechend ausführlich ausfallen. So bleibt es dieses Mal bei einer Information über die Herstellerfirma (benannt nach ihrem Gründer Craig Miller), die seit Anfang unter dem Kürzel „MG" firmiert.
In den folgenden Heften wird jeweils eines der drei angesprochenen Programme ausführlich beschrieben.
Manch einer wird einwenden, daß diese Programme doch schon in einer anderen Zeitschrift besprochen worden seien. Das ist im Prinzip richtig, jedoch war diese Besprechung recht oberflächlich und teilweise falsch. Insbesondere weist Advanced Diagnostics in der Originalversion Eigenschaften auf, die ihm gemeinhin abgesprochen werden.
Die Firma MG hat nicht nur die drei oben beschriebenen Assemblerprogramme veröffentlicht, sie lieferte auch alle drei Versionen der Betriebssystemsoftware zum CorComp-Diskcontroller (Version eins mit dem beigen Gehäuse, Version zwei im schwarzen Gehäuse und die seit Bremen erhältlichen neuen Eproms) und fertigte bis Anfang 1987 den GRAM-Kracker, die wohl beste und leistungsfähigste GRAM-Simulation für den Modulport. Seit kurzer Zeit ist es ein neues MG-Produkt, das Schlagzeilen macht, nämlich das Super X-BASIC VRS 120, ein X-BASIC Modul, das dem Mechatronic XB-II+ noch eins draufsetzt (Test in MICROpendium 9/87, Seiten 32 ff.). Daneben sollte Night-Mission nicht unerwähnt bleiben.
All diese feinen Sachen zeigen, daß von MG immer etwas Besonderes erwartet werden kann. Damit kommen wir wieder zu den eingangs erwähnten Programmen.
Wer eines der oben genannten Assembler-Highlights von MG erwirbt, wird nicht enttäuscht werden. Er erhält neben der (immer kopiergeschützten) Programmdiskette mit dem blau-weißen Aufkleber ein Handbuch, das es besonders beim Explorer in sich hat. Neben dem Ehrgeiz, mit dem Kopierschutz immer wieder etwas Neues zu bringen, erlaubt sich MG den Luxus, Manuals beizulegen, die kaum eine Frage zum Einsatzgebiet des Programms offenlassen (und aus denen dann leider abgeschrieben wird, ohne die gute Quelle zu nennen).
Alle drei Programme können entweder direkt bei Tex-Comp in USA oder beim Legio-Computer-Centre (LCC) in Holland bestellt werden – die Preise liegen bei ca. DM 60,–, was für die gebotene Leistung nicht zu viel ist.

BLICK IN DIE ZUKUNFT

In der folgenden Ausgabe der TI-REVUE wird Advanced Diagnostics im Detail besprochen werden, wobei mit einigen Falschinformationen, das Programm betreffend, aufgeräumt werden soll.
Diags, wie es auch genannt wird, ist ein leistungsfähiger Disk-Editor, der etliche Zusatzfunktionen mitbringt, die sich bei mir insbesondere beim Reparieren defekter Disketten bewährt haben.
Folgen wird die Besprechung des Explorer, nicht zu verwechseln mit The Explorer von Tarik Isani, der mit seinem Nibbler bereits weithin bekannt ist. Explorer ist eine Software-Emulation der CPU im TI-99/4A, der TMS9900. Einfach gesagt, simuliert das Programm den Ablauf anderer Programme (auch eigener), ohne sie jedoch wirklich alleine zu lassen. Es kann jederzeit der Lauf eines Programms gestoppt und Register sowie Speicher analysiert werden.
Vorläufiger Abschluß wird ein Bericht zum Diskassembler sein, dem bisher einzigen Disassembler, der sowohl Speicherinhalte als auch beliebige Files derart disassembliert, daß der erzeugte Quellcode in vielen Fällen direkt wieder reassembliert werden kann.
Dabei kann in ein File, auf Drucker oder beides simultan protokolliert werden. Zwar ist auch der Weiand Disassembler V2.1 dazu in der Lage, doch ist Diskassembler ein symbolischer Disassembler, mit allen Vorteilen, die daraus resultieren.
Na, neugierig geworden? Dann sollten Sie keineswegs die folgenden Ausgaben der TI-REVUE verpassen!

Christopher Winter (system-99-User-Group)

WO LIEGT DER DEFEKT?

Mein TI hat seinen Geist aufgegeben und zeigt statt einem gesunden Anfangsbild ein ziemliches Durcheinander. Dieser Fehler entstand beim Spielen des Parsec-Moduls.
Mitten im Spiel verschwand das Raumschiff (das Schiff des Spielers) im oberen Bildrand. Daraufhin stoppte das Programm und das Bild veränderte sich. Selbst durch einen Reset oder durch Ausschalten des Computers wurde der Fehler nicht behoben.
Beim Programmieren selbst (normales BASIC und Ex-BASIC) erschienen auf dem Bildschirm nur die Buchstaben oder Zahlen, die einen ganzzahligen ASCII-Code besitzen. Auch alle Fehlermeldungen oder sonstige Schrift veränderte der Computer so, daß die ganzzahligen Ziffern des ASCII-Codes richtig waren und die Buchstaben mit einem ungradzahligem ASCII-Code verändert wurden. Dieses geschieht so, als ob zu jedem Code 1 addiert werden würde, z.B.: Die Zahl 3 mit dem ASCII-Code 51 wird als 4 mit dem ASCII-Code 52 ausgegeben. Bei Druck auf die Zahl 4 wird diese aber richtig ausgegeben. Das passiert bei jeder anderen Taste auch: Statt einem A erscheint ein B.
Als Hobby-Elektroniker interessiert es mich natürlich, woran das liegen könnte. Leider gibt es vom TI kaum vernünftige Schaltpläne, um den Fehler vielleicht lokalisieren zu können.
Khaled Darwisch, Hannover

Ihre präzise Fehlerbeobachtung ermöglicht eine relativ sichere Aussage darüber, welches Bauteil in Ihrem TI99/4A diesen Effekt hervorruft. Vermutlich ist eines der acht VideoRAMs (TMS 4116) in der Konsole defekt. Da Ihr Rechner nicht in der Lage ist, Zeichen mit ungeraden ASCII-Codes auf dem Bildschirm darzustellen, liegt der Verdacht nahe, daß der für das niederwertigste Bit zuständige RAM-Baustein ausgefallen ist. Dies ist

nach den Schaltplänen von Texas Instruments der Baustein U102. Mir sind mehrere Fälle bekannt, wo dieser Fehler aufgetreten ist, und bei allen arbeitete der Rechner nach dem Austausch dieses Speichers wieder einwandfrei. Leider sind bei den meisten Konsolen diese RAMs eingelötet, so daß ein Austausch nur mit einiger Löterfahrung, dem richtigen Werkzeug und der nötigen Sorgfalt selbst durchgeführt werden kann. In jedem Fall empfiehlt es sich bei einer Reparatur, das neue IC nicht einzulöten, sondern an dieser Stelle einen Sockel zu benutzen und das IC zu stecken. Der Sockel und auch der TMS4116 sind nicht teuer (ca. 3 DM).

Sollte dieser Austausch jedoch nicht zum gewünschten Erfolg führen, so kann auch der Videoprozessor selbst (TMS 9929A) defekt sein. Dieser ist zwar in allen Konsolen gesokkelt, so daß ein Austausch ohne Probleme möglich ist. Leider ist dieser Videochip aber schwer zu bekommen und außerdem sehr teuer (ca. 70 DM). Fehler in anderen Bausteinen sind zwar nicht ganz auszuschließen, aber unwahrscheinlich.

Jürgen Stelter

Fill-Routine für Apesoft

Dieses Programm stellt dem Benutzer des Apesoft Grafic Generator/Interpreter Paketes den – bei diesem ansonsten sehr gelungenen Programm – schmerzlich vermißten Fill-Befehl zur Verfügung.
Der Aufruf geschieht in gleicher Weise wie bei den bereits im Apesoft-Paket enthaltenen Befehlen, d.h., auch die sogenannten "Multiple Statements" sind möglich.
Das Laden geschieht zunächst wie im Apesoft-Handbuch beschrieben, jedoch wird nach diesem Vorgang noch durch 'CALL LOAD("DSK1.FILL")' die Fill-Routine hinzugeladen.
Der Aufruf geschieht durch 'CALL LINK("FILL", X,Y)', wobei der Punkt X,Y in der zu füllenden Fläche liegen muß.
Da die Routine zusätzlichen Speicherplatz belegt, erhöht sich der im Apesoft-Handbuch auf S. 11 angegebene freizuhaltende Programmspeicherplatz von 13056 auf 13600 Bytes.

Jörg Denninghoff

```
*********************************************
*                                           *
*  FILL-FUNKTION ZUM ANBINDEN AN            *
*                                           *
*  @APESOFT GRAFIC GENERATOR/INTERPRETER    *
*                                           *
*  VERSION 1.4    COPYRIGHT 15.11.1986 BY   *
*                                           *
*  JOERG DENNINGHOFF                        *
*  5429 ALLENDORF                           *
*  HAUPTSTRASSE 38                          *
*  TEL.Ø6486/1464                           *
*                                           *
*********************************************
*                                           *
*  FUER DAS EDITOR/ASSEMBLER MODUL SIND     *
*  DIE EQUATES 'VT','ST' UND 'PT'           *
*  ANZUPASSEN                               *
*                                           *
*  BEIM XB-MODUL MUSS DER BEIM 'SIZE'       *
*  AUSGEGEBENE FREIE PROGRAMMSPEICHERPLATZ  *
*  GROESSER ALS 136ØØ BYTES SEIN !          *
*                                           *
*********************************************
*                                           *
*      R4  :STACKPOINTER                    *
*      R5  :X-RICHTUNG                      *
*      R6  :Y-RICHTUNGSANWAHL               *
*      R7  :X-BUFFER                        *
*      R8  :Y-BUFFER                        *
*      R9  :X1-BUFFER                       *
*      R1Ø :Y1-BUFFER                       *
*                                           *
*********************************************

       DEF  FILL

FAC    EQU  >834A          FLOATING POINT ACCUMULATOR
```

Weiter geht's auf Seite 51

```
VT      EQU  >24F4        APE-WERTETAFEL-BASISADRESSE              (E/A: >A3AC)
ST      EQU  >D000        STACK UND FREIER PROGRAMMSPEICHER        (E/A: >F710)
PT      EQU  >27F6        APE-UNTERPROGRAMMTAFEL BASISADRESSE      (E/A: >A552)

USRWS   EQU  VT           USER WORKSPACE
R7LB    EQU  USRWS+>F     REGISTER 7 LOWER BYTE
R8LB    EQU  USRWS+>11    REGISTER 8 LOWER BYTE

RETURN  EQU  PT           APE:RETURN TO BASIC
CFAC    EQU  PT+>22       APE:LOESCHE FAC + ERROR-REPORT-ROUTINE
PARCNT  EQU  PT+>48       APE:PARAMETERZAEHL-ROUTINE
NUMREF  EQU  PT+>96       APE:NUMREF ROUTINE
DOT     EQU  PT+>CC       APE:SETZ-/PRUEF-/LOESCH-/ROUTINE

PAR1    EQU  VT+>3C       PARAMETERANZAHLBUFFER
PAR2    EQU  VT+>3E       PARAMETERANZAHLBUFFER
XT      EQU  VT+>40       UEBERGABE ADRESSE FUER X-WERT
XTLB    EQU  XT+>1        UEBERGABE ADRESSE FUER X-WERT (LOWER BYTE)
YT      EQU  XT+>2        UEBERGABE ADRESSE FUER Y-WERT
YTLB    EQU  XT+>3        UEBERGABE ADRESSE FUER Y-WERT (LOWER BYTE)
XOFS    EQU  XT+>8        X-OFFSET
YOFS    EQU  XT+>A        Y-OFFSET
ONE     EQU  XT+>42       KONSTANTE '1'
FLAG    EQU  XT+>68       DOTFLAG: -1=SETZEN/0=LOESCHEN/1=PRUEFEN

        AORG ST
XSTACK  BSS  >0200        X-STACK
YSTACK  BSS  >0200        Y-STACK
RICHTY  DATA >0001        Y-RICHTUNG+1
        DATA >FFFF        Y-RICHTUNG-1
SUCHMU  DATA >0000        BUFFER FUER SUCHMUSTER RICHTUNG Y+1
        DATA >0000        BUFFER FUER SUCHMUSTER RICHTUNG Y+1
FILL    LWPI USRWS        LADE WORKSPACE-POINTER
        BL   @CFAC        LOESCHE FAC + ERROR-REPORT-ROUTINE
        BL   @PARCNT      PARAMETERANZAHL IN >2532
RBASIC  C    @PAR1,@PAR2  FERTIG?
        JNE  GETPAR       NEIN -->WEITERMACHEN
        B    @RETURN      RETURN TO BASIC
GETPAR  BL   @NUMREF      NUMREF
        MOV  @FAC,R7      X-WERT VON BASIC IN X
        A    @XOFS,R7     OFFSET ADDIEREN
        BL   @NUMREF      NUMREF
        NEG  @FAC         NEGIERE FAC
        MOV  @FAC,R8      Y-WERT VON BASIC IN Y
        A    @YOFS,R8     OFFSET ADDIEREN


        CLR  R4           LOESCHE STACKPOINTER
LOOP    MOV  @ONE,@FLAG   IFSET INITIALISIEREN
        MOV  R7,@XT       X-WERT ZUR UEBERGABE
        MOV  R8,@YT       Y-WERT ZUR UEBERGABE
        BL   @DOT         IFSET AUSFUEHREN
        ABS  @FLAG        IST PUNKT GESETZT?
        JNE  PULL         JA --> NEUEN AUSGANGSPUNKT VOM STACK HOLEN
        SETO @FLAG        SETTO INITIALISIEREN
        BL   @DOT         SETTO AUSFUEHREN
        CLR  @SUCHMU      SUCHMUSTER AUF '0' SETZEN
        CLR  @SUCHMU+2    SUCHMUSTER AUF '0' SETZEN
```

```
       SETO R5                     X-RICHTUNG AUF -1(LINKS) SETZEN
LOP1   MOV  R7,R9                  X IN X1
       MOV  R8,R1Ø                 Y IN Y1
LOP2ST A    R5,R9                  X-RICHTUNG ZU X1 ADDIEREN
       CLR  R6                     SCHLEIFE 2 AUF NULL SETZEN
LOP2   MOV  @ONE,@FLAG             IFSET INITIALISIEREN
       MOV  R1Ø,@YT                Y1 ZUR UEBERGABE
       A    @RICHTY(R6),@YT        Y-RICHTUNG ZU YT ADDIEREN
       MOV  R9,@XT                 X1 ZUR UEBERGABE
       S    R5,@XT                 X-RICHTUNG VON XT SUBTRAHIEREN
       BL   @DOT                   IFSET AUSFUEHREN
       C    @FLAG,@SUCHMU(R6)      IST PUNKT GLEICH SUCHMUSTER?
       JEQ  LOP2EN                 JA --> SCHLEIFE 2 WEITER
       INV  @SUCHMU(R6)            INVERTIERE SUCHMUSTER & WAR SUCHMUSTER NULL ?
       JNE  LOP2EN                 JA --> SCHLEIFE 2 WEITER
       MOVB @XTLB,@XSTACK(R4)      XT AUF X-STACK
       MOVB @YTLB,@YSTACK(R4)      YT AUF Y-STACK
       INC  R4                     STACKPOINTER ERHOEHEN
LOP2EN INCT R6                     SCHLEIFENZAEHLER 2 UM ZWEI ERHOEHEN
       CI   R6,>ØØØ4               IST SCHLEIFE 2 FERTIG?
       JNE  LOP2                   NEIN --> VON VORNE
       MOV  @ONE,@FLAG             IFSET INITIALISIEREN
       MOV  R9,@XT                 X1 ZUR UEBERGABE
       MOV  R1Ø,@YT                Y1 ZUR UEBERGABE
       BL   @DOT                   IFSET AUSFUEHREN
       ABS  @FLAG                  IST PUNKT GESETZT?
       JNE  LOP1EN                 JA --> SCHLEIFE 1 WEITER
       SETO @FLAG                  SETTO INITIALISIEREN
       BL   @DOT                   SETTO AUSFUEHREN
       JMP  LOP2ST                 SCHLEIFE 2 NEU ANFANGEN
LOP1EN INCT R5                     SCHLEIFENZAEHLER 1 UM ZWEI ERHOEHEN
       CI   R5,>ØØØ3               IST SCHLEIFE FERTIG?
       JNE  LOP1                   NEIN --> SCHLEIFE 1
       MOV  @ONE,@FLAG             IFSET INITIALISIEREN
       MOV  R7,@XT                 X ZUR UEBERGABE
       INC  R8                     Y UM EINS ERHOEHEN
       MOV  R8,@YT                 Y ZUR UEBERGABE
       BL   @DOT                   IFSET AUSFUEHREN
       ABS  @FLAG                  IST PUNKT GESETZT?
       JNE  LABEL                  JA --> NICHT AUF STACK
       MOVB @R7LB,@XSTACK(R4)      X AUF X-STACK
       MOVB @YTLB,@YSTACK(R4)      Y+1 AUF Y-STACK
       INC  R4                     STACKPOINTER ERHOEHEN
LABEL  MOV  @ONE,@FLAG             IFSET INITIALISIEREN
       DECT @YT                    UEBERGABE=Y-1
       BL   @DOT                   IFSET AUSFUEHREN
       ABS  @FLAG                  IST PUNKT GESETZT?
       JNE  PULL                   JA --> NEUEN AUSGANGSPUNKT VOM STACK HOLEN
       MOVB @R7LB,@XSTACK(R4)      X AUF X-STACK
       MOVB @YTLB,@YSTACK(R4)      Y-1 AUF Y-STACK
       INC  R4                     STACKPOINTER ERHOEHEN
PULL   DEC  R4                     STACKPOINTER VERRINGERN
       JLT  ENDE                   WENN KLEINER NULL, DANN ENDE
       MOVB @XSTACK(R4),@R7LB      X VOM X-STACK
       MOVB @YSTACK(R4),@R8LB      Y VOM Y-STACK
       JMP  LOOP                   NEUER DURCHLAUF MIT WERT VOM STACK
ENDE   B    @RBASIC                ZURUECK ZUR PARAMETERUEBERGABEROUTINE
       END
```

Lösungshilfen für The Count

In Zukunft werden in der TI-REVUE in loser Folge Hinweise zur Lösung der Scott Adams Adventures erscheinen. Am Beginn steht heute das Adventure „The Count", welches einen vor die merkwürdige Situation stellt, eines Nachmittags in einem fremden Bett miteiner Zeltstange in der Hand aufzuwachen. Schnell stellt sich heraus, daß man ausersehen wurde, die Menschheit von Graf Dracula zu befreien – und das innerhalb von drei Tagen!
Die folgenden Hinweise, eine Karte und eine Liste der für die Lösung wichtigen Gegenstände erleichtern diese schwierige Aufgabe hoffentlich ein wenig.

INHALT DER RÄUME:

Es sind nur die für die Lösung wichtigen Gegenstände aufgeführt. Gegenstände, die erst im Verlauf des Spiels auftauchen, stehen in Klammern.

Oven:	NAIL FILE
Outside:	(Postcard with PAPER CLIP), (Package with CIGARETTES)
Bed:	SHEETS, TENT STAKE
Pantry:	MATCHES, GARLIC
Workroom:	RUBBER MALLET,
Pit:	TORCH
Closet:	Vial with 3 TABLETS

EINIGE TIPS:

○ Der Aufzug (dumb waiter) stellt die Verbindung der Stockwerke untereinander her. Bewegt wird er mit RAISE DUMB und LOWER DUMB.

○ Im Keller besteht keine direkte Verbindung zwischen dem Arbeitsraum (workroom) und der Krypta. In die Krypta gelangt man nur, indem man sich mit Hilfe des Bettuchs vom Schlafraum aus in den darunterliegenden Raum (doorless room) abseilt. Dazu geht man folgendermaßen vor: GO BEDROOM, TIE SHEET, TO BED, TAKE END, GO WINDOW, PUT SHEET, CLIMB SHEET.

○ Um in die Grube (pit) im Keller zu gelangen bzw. wieder herauszukommen, leistet das Bettuch ebenfalls gute Dienste.

○ Das Klosett im Keller dient als Versteck für wichtige Gegenstände (tent stake, cigarettes), da diese einem sonst nachts entwendet werden. Die verschlossene Tür öffnet man mit dem Clip, der sich an der Postkarte befindet, die der Briefträger am ersten Tag abliefert. Nachdem man Gegenstände dort abgelegt hat, schließt man die Tür und verriegelt sie (PICK LOCK, OPEN DOOR, . . . , CLOSE DOOR, LOCK DOOR).

○ Im Klosett findet man ein Gefäß (vial) mit drei Tabletten, die sehr gut gegen Müdigkeit wirken.

○ Vor Angriffen des Vampirs schützt Knoblauch!

○ Den Ofen in der Küche kann man nur bei Nacht betreten.

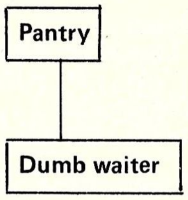
○ Den geheimen Zugang zur Krypta findet man, wenn man das Porträt von Dracula entfernt.

○ Der Sarg von Dracula wird sichtbar, wenn man in der Krypta eine Zigarette raucht. Die Zigaretten befinden sich in einem Paket, das der Briefträger am zweiten Tag abliefert.

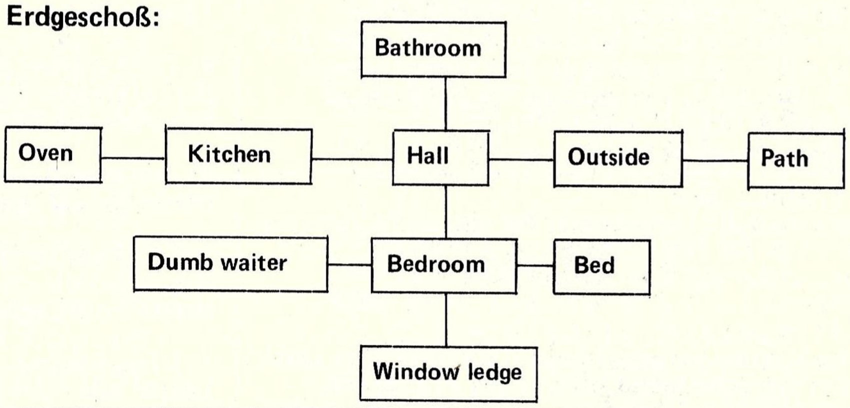
○ Das Erscheinen des Briefträgers erkennt man am Läuten der Türglocke.

○ Das Schloß an Draculas Sarg läßt sich nachts, wenn der Sarg leer ist, mit Hilfe der Feile entfernen (FILE BOLT, WITH FILE). Am Tag darauf, wenn Dracula wieder im Sarg liegt, kann man ihn mit Hilfe des Gummihammers (rubber mallet) und der Zeltstange (tent stake) unschädlich machen (KILL DRACULA).

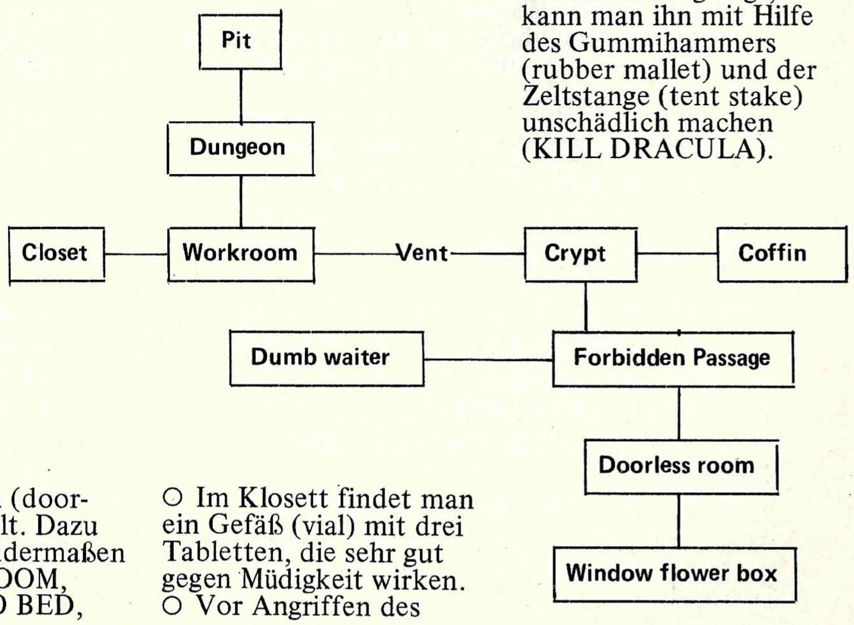
Obergeschoß:



Erdgeschoß:



Keller:



Musik- und Soundgenerator

Dieses Programm unterstützt den User beim Erstellen von Titelmelodien usw.
Mit dem Generator lassen sich:

- O Bis zu 46 Töne a vier Stimmen erzeugen und abspeichern.
- O Alle 46 Töne abspielen, egal ob sie erst eingelesen werden müssen oder bereits im internen Speicher vorhanden sind.
- O Neue Töne zwischen die abgespeicherten Töne einfügen.
- O Töne einzeln löschen.
- O Die Musikdaten abspeichern und ausdrucken.

Die Cursorsteuerung erfolgt durch die Tasten E S D X ohne das jedoch gleichzeitig die Funktionstaste gedrückt werden muß. Der gewünschte Befehl wird durch Drücken der "1" ausgeführt.

Nun zu den einzelnen Befehlen:

1) Stm 1, Stm 2, Stm 3
Hier kann man die Frequenzen der einzelnen Stimmen eingeben, z.B. für den Kammerton A = 440 Hz, oder als Tonbezeichnung, z.B. A1 bzs. Aisl A#1.

2) Stm 4
Die vierte Stimme verursacht ein Rauschen. Hier kann man die entsprechenden Codes (1 bis 8) angeben, aber ohne Minuszeichen, dieses wird automatisch eingefügt.

3) Vol 1, Vol 2, Vol 3, Vol 4
Diese Funktionen ermöglichen es, die Lautstärke der einzelnen Stimmen einzustellen, im TI-üblichen Code

4) "E"
Dieses E läßt sich auch in ein A verändern und bedeutet soviel wie "Stimme eingeschaltet" bzw. "Stimme ausgeschaltet".

5) Dauer
Die Tondauer läßt sich hiermit verändern.

6) Ton spielen
Hier läßt sich der gerade generierte Ton erproben.

7) Töne spielen
Diese Option läßt alle im Arbeitsspeicher befindliche Töne erklingen.

8) Töne spielen mit neuem Ton
Hier erklingen die Töne im Arbeitsspeicher und zusätzlich der neue Ton.

9) Ton speichern
Dieser Befehl speichert den gerade generierten Ton.

10) Ton einfügen
Hier läßt sich ein neuer Ton in das bisher abgespeicherte Ton-Register einfügen.

11) Ton löschen
Löscht alle oder nur einen Ton, je nach Wunsch.

12) Töne listen
Gibt die Töne je nach Wunsch auf den Bildschirm oder Drucker aus. Die Ausgabe ist wie bei dem Befehl CALL SOUND in Dauer, Frequenz und Lautstärke eingeteilt.

13) Töne aufnehmen
Speichert die Töne extern auf Kassette oder Diskette.

LISTINGS

```
100 ! *********************
110 ! * Musik- & Sound-    *
120 ! *          generator *
130 ! *                   *
140 ! *    Copyright by   *
150 ! *  Markus  Loerwald *
160 ! *                   *
170 ! *                   *
180 ! *                   *
190 ! * Benoetigte Geraete *
200 ! *  TI99/4A  Konsole *
210 ! *      Ext. Basic   *
220 ! *       Drucker     *
230 ! *                   *
240 ! *                   *
250 ! *                   *
260 ! *  Speicherbelegung *
270 ! *     8484 Bytes    *
280 ! *                   *
290 ! *********************
300 DATA 20
310 DATA 75,440,25,440,25,44
0,25,523,25,493,25,440,75,32
9,25,329,25,329,25,415,25,36
9,25,329,75,440,25,440
320 DATA 25,440,25,523,25,49
3,25,440,100,659,100,329
330 DATA 44
340 DATA 75,880,25,880,25,88
0,25,923,25,880,25,783,75,69
8,25,587,25,587,25,698,25,65
9,25,587
350 DATA 75,783,25,783,25,78
3,25,880,25,783,25,698,75,65
9,25,523,25,523,25,659,25,58
7,25,523
360 DATA 75,698,25,493,25,49
3,25,587,25,523,25,493,75,65
9,25,440,25,440,25,523,25,49
3,25,440
370 DATA 75,622,25,659,25,65
9,25,587,25,493,25,415,100,4
40,100,880
380 CALL SCREEN(3):: DISPLAY
 AT(3,5)ERASE ALL:"M U S I C
  M A K E R",:"     COPYRIGHT
 BY",:"     MARKUS LOERWALD",
:"":"     > TASTE"
390 FOR H=1 TO 2 :: IF H=1 T
HEN RESTORE 300 ELSE RESTORE
 330
400 READ ZNR :: FOR I=1 TO 2
 :: FOR J=1 TO ZNR :: READ A
,B :: CALL SOUND(A*10,B,2)::
 CALL KEY(0,K,S):: IF S<>0 T
HEN 430
410 NEXT J :: IF H=1 THEN RE
STORE 310 ELSE RESTORE 340
420 NEXT I :: NEXT H
430 ON WARNING NEXT :: CALL
CHAR(126,RPT$("03",8),125,RP
T$("0",12)&"FFFF",127,"FFFF"
,128,"0605050404347C3"):: CA
LL COLOR(12,5,14,13,16,14)
440 DIM L(4),T(4),ST(12,3),M
(4),DS(46),TS(4,46),LS(4,46)
,TAS(46)
450 DISPLAY AT(1,1)ERASE ALL
:"BENUTZT DU     (C)ASSETTE",
:"ODER           (D)ISKETTE",
:,"(E)NDE"
460 CALL KEY(0,K,S):: IF S=0
 THEN 460 ELSE IF K=67 THEN
G$="CS1" ELSE IF K=68 THEN G
$="DSK1." ELSE IF K=69 THEN
END ELSE 460
470 DISPLAY AT(1,1):"WILLST
DU ALTE TOENE",:"EINLADEN (J
/N)",:,"" :: CALL KEY(0,K,S)
:: IF S=0 THEN 470 ELSE IF K
=78 THEN 550 ELSE IF K<>74 T
HEN 470
480 IF G$="DSK1." THEN DISPL
AY AT(5,1):"PROGRAMMNAME ?"
:: ACCEPT AT(5,20):VA$ :: G$
="DSK1."&VA$
490 NR=0 :: OPEN #1:G$,INPUT
 ,INTERNAL,FIXED 110
500 INPUT #1:NR
510 FOR I=1 TO NR
520 INPUT #1:TAS(I),DS(I),TS
(1,I),LS(1,I),TS(2,I),LS(2,I
),TS(3,I),LS(3,I),TS(4,I),LS
(4,I)
530 NEXT I :: CLOSE #1
540 G$=SEG$(G$,1,5)
550 FOR I=1 TO 3 :: L(I)=15
:: T(I)=I*110 :: M(I)=69 ::
NEXT I :: T(I)=-4 :: L(4)=15
 :: M(I)=69 :: Z,SP,Z1=1 ::
SP1=9
560 RESTORE 1470 :: FOR I=1
TO 12 :: FOR J=1 TO 3 :: REA
D A :: ST(I,J)=A :: NEXT J :
: NEXT I
570 D=-1000 :: FOR I=1 TO 3
:: L(I)=15 :: T(I)=220+I*110
 :: NEXT I :: T(I)=-4 :: L(I
)=15
580 CALL CLEAR :: CALL SCREE
N(14)
590 FOR I=1 TO 4 :: T$=STR$(
ABS(T(I))):: L$=STR$(L(I))::
 DISPLAY AT(I*2-1,1):"STM";I
;TAB(12-LEN(T$));T(I);TAB(15
);"VOL";I;TAB(23-LEN(L$));L(
I)
600 CALL HCHAR(I*2-1,8,58)::
 CALL HCHAR(I*2-1,22,58):: C
ALL HCHAR(I*2-1,30,M(I)):: N
EXT I
```

```
610 D1=ABS(D):: T$=STR$(D1):
: DISPLAY AT(9,1):"DAUER=";T
AB(12-LEN(T$));D :: CALL HCH
AR(11,1,127,32)
620 DISPLAY AT(12,1):"TON SP
IELEN":"":"TOENE SPIELEN  MI
T NEUEM TON":"":"TON SPEICHE
RN  ZWISCHENFUEG.":"":"TON L
OESCHEN":"":"TOENE LISTEN":"
":"TOENE LADEN"
630 DISPLAY AT(24,1):"TOENE
AUFNEHMEN" :: CALL HCHAR(Z1,
SP1,128):: CALL HCHAR(20,20,
125,12):: CALL HCHAR(24,20,1
27,12):: CALL VCHAR(21,19,12
6,3)
640 CALL KEY(1,K,S):: IF S=0
 THEN 640 ELSE IF K=19 THEN
700 ELSE CALL HCHAR(Z1,SP1,3
2)
650 IF K=0 THEN Z=Z+1 :: IF
Z>12 THEN Z=Z-1 ELSE IF ST(Z
,SP)=0 THEN Z=Z-1
660 IF K=5 THEN Z=Z-1 :: IF
Z<0 THEN Z=Z+1 ELSE IF ST(Z,
SP)=0 THEN Z=Z+1
670 IF K=2 THEN SP=SP-1 :: I
F SP<0 THEN SP=SP+1 ELSE IF
ST(Z,SP)=0 THEN SP=SP+1
680 IF K=3 THEN SP=SP+1 :: I
F SP>3 THEN SP=SP-1 ELSE IF
ST(Z,SP)=0 THEN SP=SP-1
690 Z1=INT(ST(Z,SP)/32)+1 ::
 SP1=(ST(Z,SP)/32-Z1+1)*32 :
: CALL HCHAR(Z1,SP1,128):: G
OTO 640
700 IF Z1>8 THEN 790 ELSE IF
 Z1=7 AND SP<3 THEN S=1-INT(
SP/2):: SIZ=1+INT(SP/2):: VA
L$=("0123456789"):: GOTO 720
 ELSE S=0
710 IF SP=1 THEN SIZ=5 :: VA
L$="0123456789AHCDEFG#" ELSE
 IF SP=2 THEN SIZ=2 :: VAL$=
"0123456789" ELSE IF SP=3 TH
EN SIZ=1 :: VAL$="EA"
720 ACCEPT AT(Z1,SP1-1+S)SIZ
E(-SIZ)VALIDATE(VAL$):V$ ::
IF V$="" THEN 720
730 IF SP<>1 OR Z>3 THEN 750
 ELSE 1340
740 IF VAL(V$)<44000 AND VAL
(V$)>110 THEN T(Z)=VAL(V$)EL
SE 720
750 IF SP<>1 OR Z<>4 THEN 76
0 ELSE IF VAL(V$)>0 AND VAL(
V$)<9 THEN T(Z)=VAL(V$)*-1 E
LSE 720
760 IF SP<>2 THEN 770 ELSE I
F VAL(V$)>-1 AND VAL(V$)<31
THEN L(Z)=VAL(V$)ELSE 720
770 IF SP<>3 THEN 580 ELSE I
F V$="E" THEN M(Z)=69 ELSE I
F V$="A" THEN M(Z)=65
780 GOTO 580
790 ON Z-4 GOTO 800,810,930,
810,1060,1150,450,1260
800 ACCEPT AT(9,8)SIZE(-5)VA
LIDATE(DIGIT,"-+")BEEP:D ::
IF D<-4250 OR D>4250 OR D=0
THEN 800 ELSE 580
810 FOR I=1 TO 4 :: IF M(I)=
69 THEN T1=T(I):: L1=L(I)::
I=I+1 :: TA=1 :: GOTO 830
820 NEXT I :: DISPLAY AT(22,
18):"ALLE TOENE",,"   SIND A
US" :: DISPLAY AT(21,17):"
 " :: GOTO 630
830 FOR J=I TO 4 :: IF M(J)=
69 THEN T2=T(J):: L2=L(J)::
J=J+1 :: TA=2 :: GOTO 850
840 NEXT J :: GOTO 870
850 IF J>4 THEN 870 ELSE IF
M(J)=69 THEN T3=T(J):: L3=L(
J):: J=J+1 :: TA=3 :: GOTO 8
60 ELSE J=J+1 :: GOTO 850
860 IF J>4 THEN 870 ELSE IF
M(J)=69 THEN T4=T(4):: L4=L(
4):: TA=4
870 IF Z=8 THEN 950
880 ON TA GOTO 890,900,910,9
20
890 CALL SOUND(D,T1,L1):: GO
TO 640
900 CALL SOUND(D,T1,L1,T2,L2
):: GOTO 640
910 CALL SOUND(D,T1,L1,T2,L2
,T3,L3):: GOTO 640
920 CALL SOUND(D,T1,L1,T2,L2
,T3,L3,T4,L4):: GOTO 640
930 IF NR=0 THEN 810
940 FOR I=1 TO NR :: ON TAS(
I)GOSUB 1490,1500,1510,1520
:: FOR J=1 TO ABS(DS(I)/5)::
 NEXT J :: NEXT I :: IF SP=2
 THEN 810 ELSE 640
950 IF NR=46 THEN DISPLAY AT
(21,18):"SPEICHER","TOENE LA
DEN      DER TOENE",,"  IST
 VOLL" :: GOTO 630 ELSE NR=N
R+1
960 IF SP=1 THEN 1000 ELSE Z
NR=NR :: DISPLAY AT(1,1)ERAS
E ALL:"DER WIEVIELTE TON SOL
L ES":"":"SEIN ? ":"":"(ZAHL
 NICHT GROESSER ALS";NR;")"
970 ACCEPT AT(3,7)VALIDATE(D
IGIT)SIZE(2):NR :: IF NR<1 O
R NR>ZNR THEN 970
980 FOR I=ZNR TO NR+1 STEP -
1 :: TAS(I)=TAS(I-1):: DS(I)
```

LISTINGS

```
=DS(I-1):: FOR J=1 TO 4 :: T
S(J,I)=TS(J,I-1):: LS(J,I)=L
S(J,I-1):: NEXT J :: NEXT I
990 TS(2,NR)=T2 :: LS(2,NR)=
L2 :: TS(3,NR)=T3 :: LS(3,NR
)=L3 :: TS(4,J)=T4 :: LS(4,N
R)=L4
1000 TAS(NR)=TA :: DS(NR)=D
:: ON TA GOTO 1010,1020,1030
,1040
1010 TS(1,NR)=T1 :: LS(1,NR)
=L1 :: GOTO 1050
1020 TS(1,NR)=T1 :: LS(1,NR)
=L1 :: TS(2,NR)=T2 :: LS(2,N
R)=L2 :: GOTO 1050
1030 TS(1,NR)=T1 :: LS(1,NR)
=L1 :: TS(2,NR)=T2 :: LS(2,N
R)=L2 :: TS(3,NR)=T3 :: LS(3
,NR)=L3 :: GOTO 1050
1040 TS(1,NR)=T1 :: LS(1,NR)
=L1 :: TS(2,NR)=T2 :: LS(2,N
R)=L2 :: TS(3,NR)=T3 :: LS(3
,NR)=L3 :: TS(4,NR)=T4 :: LS
(4,NR)=L4
1050 IF SP=1 THEN 640 ELSE N
R=ZNR :: GOTO 580
1060 IF NR=0 THEN DISPLAY AT
(21,18):"ES SIND","TOENE LAD
EN       KEINE TOENE",,"   GE
SPEICHERT" :: GOTO 630
1070 DISPLAY AT(1,1)ERASE AL
L:"WAEHREND DES ZU LOESCHEND
EN",:"TONES TASTE DRUECKEN O
DER",:"HINTERHER NUMMER DES
TONES",:"EINGEBEN.",,:"":"TA
STE DRUECKEN"
1080 CALL KEY(0,K,S):: IF S=
0 THEN 1080 ELSE CALL CLEAR
1090 V=1 :: II=0 :: FOR I=1
TO NR :: ON TAS(I)GOSUB 1490
,1500,1510,1520 :: II=II+1 :
: DISPLAY AT(II,V):I :: IF I
/20=INT(I/20)THEN V=I/4 :: I
I=0
1100 CALL KEY(0,K,S):: IF S<
>0 THEN 1130
1110 NEXT I :: DISPLAY AT(21
,1):"GIB DEN ZU LOESCHENDEN
TON",:"EIN",,"(0=NICHTS -1=A
LLES LOESCHEN)" :: ACCEPT AT
(23,25)VALIDATE(DIGIT,"-")SI
ZE(2):I :: IF I=0 THEN 580
1120 IF I=-1 THEN NR=0 :: GO
TO 580 ELSE IF I<-1 OR I>NR
THEN 1110
1130 FOR J=I+1 TO NR :: TAS(
J-1)=TAS(J):: DS(J-1)=DS(J):
: FOR K=1 TO TAS(J):: TS(K,J
-1)=TS(K,J):: LS(K,J-1)=LS(K
,J):: NEXT K :: NEXT J
1140 NR=NR-1 :: GOTO 580
1150 IF NR=0 THEN DISPLAY AT
(21,18):"ES SIND","TOENE LAD
EN       KEINE TOENE",,"   GE
SPEICHERT" :: GOTO 630
1160 DISPLAY AT(1,1)ERASE AL
L:"SOLL ICH DIE TOENE":"":">
A AUF DEM BILDSCHIRM LISTEN"
:"":">B AUSDRUCKEN"
1170 CALL KEY(0,K,S):: IF S=
0 THEN 1170 ELSE IF K=65 THE
N 1210 ELSE IF K<>66 THEN 11
70
1180 OPEN #1:"RS232.BA=9600"
,FIXED 80
1190 PRINT #1:"NR  DAUER   S
TM     VOL" :: FOR I=1 TO NR
:: PRINT #1:I;TAB(5);DS(I);:
: FOR J=1 TO TAS(I):: PRINT
#1:TAB(12);TS(J,I);TAB(20);L
S(J,I):: NEXT J :: NEXT I
1200 CLOSE #1 :: GOTO 580
1210 CALL CLEAR :: PRINT "NR
  DAUER   STM     VOL" :: FOR
 I=1 TO NR :: PRINT I;TAB(5)
;DS(I);:: FOR J=1 TO TAS(I):
: PRINT TAB(12);TS(J,I);TAB(
20);LS(J,I):: NEXT J
1220 IF I/5=INT(I/5)THEN PRI
NT :: DISPLAY AT(24,1):"> TA
STE 1=ARBEITSBILD" ELSE 1240
1230 CALL KEY(0,K,S):: IF S=
0 THEN 1230 ELSE IF K=49 THE
N 580 ELSE DISPLAY ERASE ALL
:"NR  DAUER   STM     VOL"
1240 NEXT I :: DISPLAY "> TA
STE"
1250 CALL KEY(0,K,S):: IF S=
0 THEN 1250 ELSE 580
1260 IF NR=0 THEN DISPLAY AT
(21,18):"ES SIND","TOENE LAD
EN       KEINE TOENE",,"   GE
SPEICHERT" :: GOTO 630
1270 IF G$="DSK1." THEN DISP
LAY AT(1,1)ERASE ALL:"PROGRA
MMNAME ?" :: ACCEPT AT(1,20)
:VA$ :: G$="DSK1."&VA$
1280 OPEN #1:G$,OUTPUT,INTER
NAL,FIXED 110
1290 PRINT #1:NR
1300 FOR I=1 TO NR
1310 PRINT #1:TAS(I),DS(I),T
S(1,I),LS(1,I),TS(2,I),LS(2,
I),TS(3,I),LS(3,I),TS(4,I),L
S(4,I)
1320 NEXT I
1330 CLOSE #1 :: GOTO 580
1340 J=LEN(V$):: FOR I=1 TO
J :: VA$=SEG$(V$,I,1):: IF A
SC(VA$)>64 THEN 1360
1350 NEXT I :: GOTO 740
1360 A=2^(1/12):: ON ASC(VA$
)-64 GOTO 1370,32767,1380,13
```

```
90,1400,1410,1420,1430
1370 B=0 :: GOTO 1440
1380 B=3 :: GOTO 1440
1390 B=5 :: GOTO 1440
1400 B=7 :: GOTO 1440
1410 B=8 :: GOTO 1440
1420 B=10 :: GOTO 1440
1430 B=2
1440 IF I+1>J THEN 1460
1450 I=I+1 :: VA$=SEG$(V$,I,
1):: IF VA$="#" THEN B=B+1 :
: GOTO 1440 ELSE IF ASC(VA$)
>48 AND ASC(VA$)<55 THEN B=B
+12*(ASC(VA$)-48)ELSE 720
1460 T(Z)=INT(110*A^B):: GOT
O 590
1470 DATA 9,23,29,73,87,93,1
37,151,157,204,215,221,265,0
,0,366,0,0,432,447,0,496,511
,0,559,0,0,623,0,0
1480 DATA 686,0,0,754,0,0
1490 CALL SOUND(DS(I),TS(1,I
),LS(1,I)):: RETURN
1500 CALL SOUND(DS(I),TS(1,I
),LS(1,I),TS(2,I),LS(2,I))::
 RETURN
1510 CALL SOUND(DS(I),TS(1,I
),LS(1,I),TS(2,I),LS(2,I),TS
(3,I),LS(3,I)):: RETURN
1520 CALL SOUND(DS(I),TS(1,I
),LS(1,I),TS(2,I),LS(2,I),TS
(3,I),LS(3,I),TS(4,I),LS(4,I
)):: RETURN
```

LISTINGS

Zahlen mit dem Speech Synthesizer

Dieses kleine Programm ermöglicht dem Sprachsynthesizer von Texas Instruments größere Zahlen, z.B. 897 , nicht Zahl für Zahl hintereinander zu sprechen, sondern den Wert in Hundertern, Zehnern und Einern richtig wiederzugeben. Hier würde es dann lauten: 'eight hundred ninety seven'. Da dieses Programm nicht sehr groß ist, kann man es ohne Probleme auch in andere Programme mit einbauen.

Titelbild für Video Movie

Bei diesem Programm handelt es sich um einen Vorspann für eigene Videoaufnahmen. Das Programm ist in TI-Basic geschrieben und natürlich nur in Verbindung mit einem Videorecorder sinnvoll.
Mit dem Programm kann man Selbstgefilmtes numerieren und mit Filmtitel und Inhaltsangabe versehen. Realisiert wird das Ganze, indem man den TI an einen Videorecorder anschließt und den Vorspann vor und das Schlußbild nach den eigenen Aufnahmen aufzeichnet.

```
100 ! **********************
110 ! *                    *
120 ! * Sprachenausgabeprg.*
130 ! *                    *
140 ! *    Copyright by    *
150 ! *   Knut  Meissner   *
160 ! *                    *
170 ! *                    *
180 ! *                    *
190 ! * Benoetigte Geraete *
200 ! *   TI99/4A  Konsole *
210 ! *     Ext. Basic     *
220 ! * Speech Synthesizer *
230 ! *                    *
240 ! *                    *
250 ! *                    *
260 ! *   Speicherbelegung *
270 ! *     1000 Bytes     *
280 ! *                    *
290 ! **********************
300 INPUT A
310 A1$=STR$(A):: V,M=LEN(A1
$):: IF A=0 THEN C$="ZERO" :
: GOTO 450
320 FOR F=1 TO V
330 A$=(SEG$(A1$,F,1))
340 ON V GOTO 370,380,420
350 B1$,B$="" :: NEXT F
360 B$,B1$,C$="" :: GOTO 300
370 RESTORE 470 :: FOR I=0 T
O (VAL(A$)-1):: READ B1$ ::
NEXT I :: C$=C$&"+"&B1$ :: G
OTO 450
380 IF VAL(A$)=1 THEN 390 EL
SE 410
390 Z=VAL(SEG$(A1$,M,1))
400 RESTORE 480 :: FOR I=0 T
O Z :: READ B1$ :: NEXT I ::
C$=C$&"+"&B1$ :: V=V-1 :: GO
TO 450
410 RESTORE 490 :: FOR I=1 T
O VAL(A$):: READ B1$ :: NEXT
I :: C$=C$&"+"&B1$ :: IF
SEG$(A1$,1,1)="0" THEN 450
ELSE V=V-1 :: GOTO 350
420 RESTORE 470 :: FOR I=1
TO VAL(A$):: READ B$ 430
NEXT I :: B1$=B$&"+HUNDRED"
:: C$=B1$ 440 V=V-1 :: GOTO
350
450 CALL SAY(C$):: C$=""
460 GOTO 300
470 DATA ONE,TWO,THREE,FOUR,
FIVE,SIX,SEVEN,EIGHT,NINE
480 DATA TEN,ELEVEN,TWELVE,
THIRTEEN,FOURTEEN,FIFTEEN,
SIX+TEN,SEVEN+TEN,EIGHT+TEN
,NINE+TEN
490 DATA TEN,TWENTY,THIRTY,
FORTY,FIFTY,SIXTY,SEVENTY,
EIGHTY,NINETY
```

LISTINGS

```
100 REM********************
110 REM*Titelbild f. Video*
120 REM*                  *
130 REM*  Copyright  by   *
140 REM* JUERGEN    HORST *
150 REM*                  *
160 REM*                  *
170 REM*Benoetigte Geraete*
180 REM* TI99/4A  Konsole *
190 REM* Video-Recorder   *
200 REM*                  *
220 REM*                  *
230 REM* Speicherbelegung *
240 REM*    8400 Bytes    *
250 REM********************
260 REM
270 CALL CLEAR
280 GOTO 1560
290 CALL CHAR(149,"80C0E0F0F
8FCFEFF")
300 CALL CHAR(150,"FF7F3F1F0
F070301")
310 CALL CHAR(151,"0103070F1
F3F7FFF")
320 CALL CHAR(152,"FFFEFCF8F
0E0C080")
330 CALL CHAR(128,"0F0F0F0F0
F0F0F0F")
340 CALL CHAR(129,"0000FFFFF
FFF0000")
350 CALL CHAR(130,"F0F0F0F0F
0F0F0F0")
360 CALL CHAR(131,"0000070F0
F0F0F0F")
370 CALL CHAR(133,"F0F0F0F0F
0E00000")
380 CALL CHAR(132,"0000E0F0F
0F0F0F0")
390 CALL CHAR(134,"0F0F0F0F0
F070000")
400 CALL CHAR(135,"0F0F0F070
0000000")
410 CALL CHAR(136,"000000000
70F0F0F")
420 CALL CHAR(137,"0F0F0F070
70F0F0F")
430 CALL CHAR(138,"F0F0F0E00
0000000")
440 CALL CHAR(139,"00000000E
0F0F0F0")
450 CALL CHAR(140,"F0F0F0E0E
0F0F0F0")
460 CALL CLEAR
470 CALL SCREEN(6)
480 CALL COLOR(15,12,6)
490 CALL COLOR(6,12,12)
500 CALL COLOR(2,16,7)
510 CALL COLOR(16,12,6)
520 CALL COLOR(13,2,12)
530 CALL COLOR(14,2,12)
540 CALL COLOR(8,12,12)
550 CALL VCHAR(2,3,43,22)
560 CALL HCHAR(2,4,43,26)
570 CALL HCHAR(23,4,43,26)
580 CALL VCHAR(2,30,43,22)
590 REM
600 CALL VCHAR(4,6,72,3)
610 CALL VCHAR(4,9,72,3)
620 CALL HCHAR(7,6,150,1)
630 CALL HCHAR(7,7,149,1)
640 CALL HCHAR(8,7,150,1)
650 CALL HCHAR(8,8,152,1)
660 CALL HCHAR(7,8,151,1)
670 CALL HCHAR(7,9,152,1)
680 CALL VCHAR(4,12,72,5)
690 CALL VCHAR(4,15,72,5)
700 CALL VCHAR(5,17,72,3)
710 CALL HCHAR(4,16,72,1)
720 CALL HCHAR(8,16,72,1)
730 CALL HCHAR(4,17,149,1)
740 CALL HCHAR(8,17,152,1)
750 CALL VCHAR(4,20,72,5)
760 CALL HCHAR(4,21,72,2)
770 CALL HCHAR(8,21,72,2)
780 CALL HCHAR(6,21,72,1)
790 CALL HCHAR(4,26,72,1)
800 CALL HCHAR(8,26,72,1)
810 CALL VCHAR(5,25,72,3)
820 CALL VCHAR(5,27,72,3)
830 CALL HCHAR(4,25,151,1)
840 CALL HCHAR(4,27,149,1)
850 CALL HCHAR(8,25,150,1)
860 CALL HCHAR(8,27,152,1)
870 CALL VCHAR(11,6,72,4)
880 CALL VCHAR(11,9,72,4)
890 CALL HCHAR(10,6,149,1)
900 CALL HCHAR(11,7,149,1)
910 CALL HCHAR(12,7,150,1)
920 CALL HCHAR(12,8,152,1)
930 CALL HCHAR(11,8,151,1)
940 CALL HCHAR(10,9,151,1)
950 CALL VCHAR(11,12,72,3)
960 CALL VCHAR(11,14,72,3)
970 CALL HCHAR(10,13,72,1)
980 CALL HCHAR(14,13,72,1)
990 CALL HCHAR(10,12,151,1)
1000 CALL HCHAR(10,14,149,1)
1010 CALL HCHAR(14,12,150,1)
1020 CALL HCHAR(14,14,152,1)
1030 CALL VCHAR(10,17,72,3)
1040 CALL VCHAR(10,20,72,3)
1050 CALL HCHAR(13,17,150,1)
1060 CALL HCHAR(13,18,149,1)
1070 CALL HCHAR(14,18,150,1)
1080 CALL HCHAR(14,19,152,1)
1090 CALL HCHAR(13,19,151,1)
1100 CALL HCHAR(13,20,152,1)
1110 REM
1120 CALL VCHAR(10,23,72,5)
1130 CALL VCHAR(10,26,72,5)
```

LISTINGS

```
1140 CALL HCHAR(10,27,72,2)
1150 CALL HCHAR(12,27,72,1)
1160 CALL HCHAR(14,27,72,2)
1170 CALL VCHAR(17,6,72,4)
1180 CALL VCHAR(16,9,72,4)
1190 CALL HCHAR(16,6,149,1)
1200 CALL HCHAR(17,7,149,1)
1210 CALL HCHAR(18,7,150,1)
1220 CALL HCHAR(18,8,149,1)
1230 CALL HCHAR(19,8,150,1)
1240 CALL HCHAR(20,9,150,1)
1250 CALL VCHAR(16,12,72,5)
1260 CALL HCHAR(16,13,72,2)
1270 CALL HCHAR(18,13,72,2)
1280 CALL HCHAR(17,15,72,1)
1290 CALL HCHAR(19,13,150,1)
1300 CALL HCHAR(16,15,149,1)
1310 CALL HCHAR(18,15,152,1)
1320 CALL HCHAR(20,14,150,1)
1330 CALL HCHAR(19,14,149,1)
1340 CALL HCHAR(20,15,149,1)
1350 CALL HCHAR(17,18,72,1)
1360 CALL HCHAR(19,18,72,1)
1370 CALL VCHAR(16,21,88,5)
1380 CALL VCHAR(16,22,88,5)
1390 CALL VCHAR(16,23,88,5)
1400 CALL VCHAR(16,25,88,5)
1410 CALL VCHAR(16,26,88,5)
1420 CALL VCHAR(16,27,88,5)
1430 IF A$="1" THEN 1790
1440 IF A$="2" THEN 1850
1450 IF A$="3" THEN 1970
1460 IF A$="4" THEN 2010
1470 IF A$="5" THEN 2060
1480 IF A$="6" THEN 2180
1490 IF A$="7" THEN 2240
1500 IF A$="8" THEN 2270
1510 IF A$="9" THEN 2290
1520 IF A$="0" THEN 2370
1530 IF B$="1" THEN 2500
1540 IF B$="8" THEN 2890
1550 GOTO 2890
1560 CALL CLEAR
1570 PRINT "TITELBILD FUER V
IDEO MOVIE"
1580 PRINT
1590 PRINT "VON J.M.HORST"
1600 PRINT
1610 PRINT
1620 PRINT
1630 INPUT "ERSTE ZAHL ?":A$
1640 PRINT
1650 INPUT "ZWEITE ZAHL ?":B
$
1660 PRINT
1670 INPUT "FILMTITEL (BIS 2
 ZEILEN)    ?":C$
1680 PRINT
1690 INPUT "AUFNAHMEZEIT (1
ZEILE)      ?":D$
1700 PRINT
1710 INPUT "SCHAUSPIELER (3
ZEILEN)     ?":E$
1720 PRINT
1730 INPUT "BEMERKUNGEN (4 Z
EILEN)      ?":F$
1740 CALL CLEAR
1750 CALL SCREEN(11)
1760 FOR K=1 TO 1000 STEP 2
1770 NEXT K
1780 GOTO 290
1790 CALL HCHAR(16,23,132,1)
1800 CALL HCHAR(17,23,130,1)
1810 CALL HCHAR(18,23,140,1)
1820 CALL HCHAR(19,23,130,1)
1830 CALL HCHAR(20,23,133,1)
1840 GOTO 1530
1850 CALL HCHAR(16,21,131,1)
1860 CALL HCHAR(16,22,129,1)
1870 CALL HCHAR(16,23,132,1)
1880 CALL HCHAR(17,23,130,1)
1890 CALL HCHAR(18,23,138,1)
1900 CALL HCHAR(18,22,129,1)
1910 CALL HCHAR(18,21,136,1)
1920 CALL HCHAR(19,21,128,1)
1930 CALL HCHAR(20,21,134,1)
1940 CALL HCHAR(20,22,129,1)
1950 CALL HCHAR(20,23,133,1)
1960 GOTO 1530
1970 CALL HCHAR(16,22,129,1)
1980 CALL HCHAR(18,22,129,1)
1990 CALL HCHAR(20,22,129,1)
2000 GOTO 1790
2010 CALL HCHAR(16,21,131,1)
2020 CALL HCHAR(17,21,128,1)
2030 CALL HCHAR(18,21,135,1)
2040 CALL HCHAR(18,22,129,1)
2050 GOTO 1790
2060 CALL HCHAR(16,22,129,1)
2070 CALL HCHAR(16,21,131,1)
2080 CALL HCHAR(17,21,128,1)
2090 CALL HCHAR(18,21,135,1)
2100 CALL HCHAR(18,22,129,1)
2110 CALL HCHAR(18,23,139,1)
2120 CALL HCHAR(19,23,130,1)
2130 CALL HCHAR(20,23,133,1)
2140 CALL HCHAR(20,22,129,1)
2150 CALL HCHAR(20,21,134,1)
2160 IF A$="6" THEN 2190
2170 GOTO 1530
2180 GOTO 2060
2190 CALL HCHAR(16,22,88,1)
2200 CALL HCHAR(19,21,128,1)
2210 CALL HCHAR(18,21,137,1)
2220 CALL HCHAR(16,23,88,1)
2230 GOTO 1530
2240 CALL HCHAR(16,22,129,1)
2250 CALL HCHAR(16,21,131,1)
2260 GOTO 1790
2270 CALL HCHAR(18,22,129,1)
2280 GOTO 2370
```

LISTINGS

```
2290 CALL HCHAR(16,21,131,1)
2300 CALL HCHAR(16,22,129,1)
2310 CALL HCHAR(17,21,128,1)
2320 CALL HCHAR(18,21,135,1)
2330 CALL HCHAR(18,22,129,1)
2340 CALL HCHAR(20,21,134,1)
2350 CALL HCHAR(20,22,129,1)
2360 GOTO 1790
2370 CALL HCHAR(16,23,132,1)
2380 CALL HCHAR(17,23,130,1)
2390 CALL HCHAR(18,23,140,1)
2400 CALL HCHAR(19,23,130,1)
2410 CALL HCHAR(20,23,133,1)
2420 CALL HCHAR(16,21,131,1)
2430 CALL HCHAR(17,21,128,1)
2440 CALL HCHAR(18,21,137,1)
2450 CALL HCHAR(16,22,129,1)
2460 CALL HCHAR(19,21,128,1)
2470 CALL HCHAR(20,22,129,1)
2480 CALL HCHAR(20,21,134,1)
2490 GOTO 1530
2500 CALL HCHAR(16,27,132,1)
2510 CALL HCHAR(17,27,130,1)
2520 CALL HCHAR(18,27,140,1)
2530 CALL HCHAR(19,27,130,1)
2540 CALL HCHAR(20,27,133,1)
2550 GOTO 3160
2560 CALL HCHAR(17,25,88,1)
2570 CALL HCHAR(19,27,88,1)
2580 CALL HCHAR(18,25,136,1)
2590 CALL HCHAR(18,27,138,1)
2600 GOTO 3160
2610 CALL HCHAR(17,25,88,1)
2620 CALL HCHAR(18,25,88,1)
2630 CALL HCHAR(19,25,88,1)
2640 CALL HCHAR(16,25,88,1)
2650 CALL HCHAR(20,25,88,1)
2660 GOTO 3160
2670 CALL HCHAR(16,26,88,1)
2680 CALL HCHAR(19,25,88,1)
2690 CALL HCHAR(20,25,88,1)
2700 CALL HCHAR(20,26,88,1)
2710 CALL HCHAR(18,25,135,1)
2720 GOTO 3160
2730 CALL HCHAR(16,27,88,1)
2740 CALL HCHAR(17,27,88,1)
2750 CALL HCHAR(19,25,88,1)
2760 CALL HCHAR(18,25,135,1)
2770 CALL HCHAR(18,27,139,1)
2780 GOTO 3160
2790 CALL HCHAR(16,26,88,1)
2800 CALL HCHAR(16,27,88,1)
2810 CALL HCHAR(17,27,88,1)
2820 CALL HCHAR(18,27,139,1)
2830 GOTO 3160
2840 CALL HCHAR(17,25,88,1)
2850 CALL HCHAR(18,25,88,2)
2860 CALL HCHAR(19,25,88,1)
2870 CALL HCHAR(20,25,88,2)
2880 GOTO 3160
2890 CALL HCHAR(16,25,131,1)
2900 CALL HCHAR(16,26,129,1)
2910 CALL HCHAR(16,27,132,1)
2920 CALL HCHAR(17,27,130,1)
2930 CALL HCHAR(18,27,140,1)
2940 CALL HCHAR(19,27,130,1)
2950 CALL HCHAR(20,27,133,1)
2960 CALL HCHAR(20,26,129,1)
2970 CALL HCHAR(20,25,134,1)
2980 CALL HCHAR(19,25,128,1)
2990 CALL HCHAR(18,25,137,1)
3000 CALL HCHAR(17,25,128,1)
3010 CALL HCHAR(18,26,129,1)
3020 IF B$="9" THEN 3110
3030 IF B$="0" THEN 3140
3040 IF B$="7" THEN 2840
3050 IF B$="6" THEN 2790
3060 IF B$="5" THEN 2730
3070 IF B$="4" THEN 2670
3080 IF B$="3" THEN 2610
3090 IF B$="2" THEN 2560
3100 GOTO 3160
3110 CALL HCHAR(18,25,135,1)
3120 CALL HCHAR(19,25,88,1)
3130 GOTO 3160
3140 CALL HCHAR(18,26,88,1)
3150 GOTO 3160
3160 FOR I=1 TO 3000 STEP 2
3170 NEXT I
3180 CALL CLEAR
3190 CALL SCREEN(2)
3200 CALL COLOR(1,6,1)
3210 CALL COLOR(2,6,1)
3220 CALL COLOR(3,6,1)
3230 CALL COLOR(4,6,1)
3240 CALL COLOR(5,6,1)
3250 CALL COLOR(6,6,1)
3260 CALL COLOR(7,6,1)
3270 CALL COLOR(8,6,1)
3280 CALL COLOR(9,6,1)
3290 PRINT "FILMTITEL :"
3300 PRINT
3310 PRINT C$
3320 PRINT
3330 PRINT "AUFNAHMEZEIT :"
3340 PRINT
3350 PRINT D$
3360 PRINT
3370 PRINT "SCHAUSPIELER :"
3380 PRINT
3390 PRINT E$
3400 PRINT
3410 PRINT "BEMERKUNGEN :"
3420 PRINT
3430 PRINT F$
3440 PRINT
3450 PRINT
3460 FOR X=1 TO 7000 STEP 2
3470 NEXT X
3480 CALL CLEAR
3490 CALL SCREEN(16)
```

```
3500 FOR J=0 TO 2000 STEP 2
3510 NEXT J
3520 CALL CLEAR
3530 CALL SCREEN(5)
3540 CALL COLOR(2,16,2)
3550 CALL COLOR(15,9,1)
3560 CALL COLOR(16,9,1)
3570 CALL COLOR(6,9,9)
3580 CALL VCHAR(5,4,43,12)
3590 CALL VCHAR(5,28,43,12)
3600 CALL HCHAR(5,5,43,23)
3610 CALL HCHAR(16,5,43,23)
3620 CALL VCHAR(8,6,72,5)
3630 CALL HCHAR(8,7,72,2)
3640 CALL HCHAR(10,7,72,1)
3650 CALL HCHAR(12,7,72,2)
3660 CALL VCHAR(9,11,72,4)
3670 CALL HCHAR(8,11,149,1)
3680 CALL HCHAR(9,12,149,1)
3690 CALL HCHAR(10,12,150,1)
3700 CALL HCHAR(10,13,149,1)
3710 CALL HCHAR(11,13,150,1)
3720 CALL HCHAR(12,14,150,1)
3730 CALL VCHAR(8,14,72,4)
3740 CALL VCHAR(8,17,72,5)
3750 CALL VCHAR(9,19,72,3)
3760 CALL HCHAR(8,18,72,1)
3770 CALL HCHAR(12,18,72,1)
3780 CALL HCHAR(12,19,152,1)
3790 CALL HCHAR(8,19,149,1)
3800 CALL VCHAR(8,22,72,5)
3810 CALL HCHAR(8,23,72,2)
3820 CALL HCHAR(10,23,72,1)
3830 CALL HCHAR(12,23,72,2)
3840 FOR M=1 TO 6000 STEP 2
3850 NEXT M
3860 CALL CLEAR
3870 CALL SCREEN(11)
3880 FOR G=1 TO 2000 STEP 2
3890 NEXT G
3900 CALL COLOR(1,13,1)
3910 CALL COLOR(2,13,1)
3920 CALL COLOR(3,13,1)
3930 CALL COLOR(4,13,1)
3940 CALL COLOR(5,13,1)
3950 CALL COLOR(6,13,1)
3960 CALL COLOR(7,13,1)
3970 CALL COLOR(8,13,1)
3980 CALL COLOR(9,13,1)
3990 CALL SCREEN(2)
4000 PRINT
4010 PRINT
4020 PRINT
4030 PRINT "WAEHLE :"
4040 PRINT
4050 PRINT
4060 PRINT "VORGANG WIEDERHO
LEN --1--"
4070 PRINT
4080 PRINT "NEUE EINGABEN
 --2--"
```

LISTINGS

```
4090 PRINT
4100 PRINT "PROGRAMM BEENDEN
 --3--"
4110 PRINT
4120 PRINT
4130 PRINT
4140 PRINT
4150 INPUT " DIE WAHL LAUTET
 ? ":G$
4160 PRINT
4170 PRINT
4180 PRINT
4190 PRINT
4200 IF G$="1" THEN 290
4210 IF G$="2" THEN 1560
4220 IF G$="3" THEN 4230
4230 CALL CLEAR
4240 PRINT
4250 PRINT "DAS WAR'S"
4260 PRINT
4270 PRINT
4280 PRINT
4290 PRINT
```

TI BÖRSE

Suche Sprachsynth., 3fach-Modulexpander, 32K f. die Box, E/A-Anl. dt., Slimline-LW, Anl. f. Turbo-Pascal 99, Ti99/4A-Konsole günstig zu kaufen. Martin Lutz, Jenaerstr. 43, 3180 Wolfsburg

---

Stat. RAMs UPD43256 (32K x8) v. NEC zu verkaufen. Preis je 20,– DM (habe aus Preisgründen mehr gekauft als benötigt). Tel. 06257/83247

Zu verk.: TI99/4A, Rec.-Kabel, Ext. Basic-Mod., Basic-Kurse, Lit., Schnittst. (orig. TI) m. eigener Stromvers. (2 x V24, 1 x Centr.), Druckerkabel, alles neuwertig, teilweise orig.-verp. Tel. 0241/525204

TI99/4A: neu 100,– DM, gebr. 50,– DM. Peri-Box, 2 Disk-Dr., Karte, Mod. 280,– DM; Epson MX80 400,– DM; Mini Mem, orig., Handb. dt., 50,– DM; RS232 150,– DM; 32K-Erw. 150,– DM; Color TV, 37 CM, inkl. RGB-Interf. 400,– DM; div. Module a 10,– DM u. div. Bücher. Abholpreise. Josef Schaller, Glattalstr. 127, CH-8052-Zürich

---

Achtung, TI-User. Wer hat das Modul IBM-Connection gekauft u. bekommt es nicht zum Laufen? Meldet Euch bei mir. Wolfgang Bürger, Birther Str. 35, 5620 Velbert 1

Disc-Contr., RS232 u. 1 LW 850,– DM; 1 Sprachsynth. 80,– DM; 1 Mod.-Exp. 3fach 90,– DM; 1 Druckerkab. f. GP-100 30,– DM; 1 Kass.-Rec. m. Kabel 50,– DM. Module: TI-Basic 100,– DM; Buchungsj. 100,– DM; Schachm. 70,– DM; Stat. 70,– DM; Pers. Rec. Keep. 30,– DM; Pers. Rep. Gen. 40,– DM; Disk-Man. 50,– DM. Orig.-Prog.: Versandl. 60,– DM; Lagerverw. 60,– DM; Rechn.-St. 60,– DM; Basic-Rout. 1,2,3 70,– DM; Finanzber. 10,– DM; Bas. f. Anf. 10,– DM; TI-Writer dt. 150,– DM. Fachbücher: 99 Special 1, 99 Special 2, TI-Basic v. Becker, Prog. f. TI99/4A, Spielen, Lernen, Arbeiten m. TI99/4A, 2 x TI-Basic-Ex., zus. 200,– DM. Außerdem ca. 300 Prog. auf Disk. u. 100 Prog. auf Kass., TI REVUE v. 84-87. Alles zusa. zum Top-Preis v. 2.000,– DM. Peter Thor, Steinauer Str. 13, 6000 Frankfurt 61, Tel. 069/414539

Verk. TI99/4A, P-Box, Contr., 2 LW RS232, 32K, Speech-Synth. u. Softw. Preis: 1.300,– DM. Auch einzeln. Tel. 06081/7529

TI99 kompl.: P-Box (RS232, 2 Slimline int.), 32K, Speech-Synth., XB II Plus, E/A, D-Man., TE II, D.-Fixer, C-Compiler, Color-Mon., 2. Kons. (TI99), orig. Floppy, Lit. u. Disks, VB 1.800,– DM. Tel. 040/2504473 ab 19 Uhr

---

Die Programmierung einer DSR-Teil III

Heute möchte ich Ihnen die Routinen für OPEN, CLOSE und PRINT vorstellen. Die dateiverarbeitenden Routinen sind naturgemäß etwas aufwendiger als die programmverarbeitenden, da sie mit den verschiedensten Formaten hantieren müssen. Zunächst ist es wieder wichtig, zu wissen, wie die drei RAM-Segmente aussehen, wenn sie nun eine Datei enthalten:

Bytes 0 & 1	5AA5	RAM-Segment enthält ein Basic-File
Byte 2	..	>00: FIX/DIS >10: VAR/DIS >08: FIX/INT >18: VAR/INT
Byte 3	..	(max.) Satzlänge
Bytes 4 & 5	....	End-of-File Pointer
Bytes 6 & 7	....	Current File Entry Pointer

Bytes 8 bis Segmentende: Datenfile.

Sowohl der End-of-File Pointer als auch der Current-File Entry Pointer sind relative Pointer. Bei einer VARIABLE-Datei muß zu ihnen erst die Startadresse des zugehörigen Datenfiles addiert werden. Der Current File Entry Pointer zeigt dann auf die Adresse, ab der der folgende Datensatz geschrieben wird, und der End-of-File Pointer zeigt auf das Byte nach dem letzten Datensatz. Bei FIXED-Dateien enthalten beide Pointer nur REC-Nummern. Der Current File Entry Pointer zeigt dann an, welcher Datensatz als nächstes geschrieben wird und der End-of-File Pointer enthält die Nummer plus 1 des höchsten Datensatzes in der Datei.

Betrachten wir uns die Routine OPEN. Je nachdem, ob die Anweisung auf INPUT, OUTPUT, UPDATE oder APPEND lautet, muß sie verschiedene Dinge abarbeiten:

INPUT Wenn ein File im Speicher ist, dann muß es das gleiche Format (Display oder Internal, Fixed oder Variable) und die gleiche Satzlänge haben, sonst gibt's Errors ('Bad Attribute'). Ist kein File im Speicher, dann gibt's sowieso einen Error ('File Error').

OUTPUT Initialisiert den Speicher grundsätzlich neu.

UPDATE Ist ein File im Speicher, dann genau wie INPUT, ansonsten wird neu initalisiert.

APPEND Wie UPDATE.

Die Routine sieht dementsprechend so aus:

```
G61E1  ST    §>834A,§>834B    Statusbyte des nach >834A kopierten PABs
G61E4  AND   §>834A,>06       Bits IN/OUT/UPD./APP. abspalten
G61E7  DCEQ  §>0000(§>835A),>5AA5  Bereich muß mit >5AA5 starten.
G61EE  BS    GROM§>61FA             Tut es: Sprung

G61F0  CEQ   §>834A,>04        INPUT ?
G61F3  BS    GROM§>61D0        Ja: 'File Error'
G61F5  CALL  GROM§>646B        Nein: Initialisierung
G61F8  BR    GROM§>6229        weiter

G61FA  CEQ   §>834A,>02      OUTPUT ?
G61FD  BR    GROM§>6204      Nein: Sprung
G61FF  CALL  GROM§>646B      Ja: Initialisierung
G6202  BR    GROM§>6229      weiter

G6204  ST    §>834A,§>834B   Statusbyte erneut kopieren
G6207  AND   §>834A,>18      Bits FIX/VAR und DIS/INT abspalten
G620A  CEQ   §>834A,§>0002(§>835A) Vergleich mit dem Identifier
                             des gespeicherten Files
G6210  BR    GROM§>61B8      Ungleich: 'Bad Attribute'
G6212  CZ    §>834E          Satzlänge gleich null ?
G6214  BR    GROM§>6221      Nein: weiter zum Satzlängenvergleich
G6216  ST    VDP§>FFFA(§>8358),  Ja: Länge des gespeicherten Files
             §>0003(§>835A)           als Default in den PAB
G621F  BR    GROM§>6229      weiter
G6221  CEQ   §>834E,§>0003(§>835A)  Vergleich der Satzlänge mit
                                    der des gespeicherten Files
G6227  BR    GROM§>61B8      Ungleich: 'Bad Attribute'
```

Nun müssen noch der Current File Entry Pointer und der End-of-File Pointer gesetzt werden.

INPUT War ein File im Speicher, dann Current File Entry Pointer nullen und End-of-File Pointer unverändert, ansonsten werden beide Null gesetzt.

OUTPUT Beide Pointer werden Null gesetzt

UPDATE Wie INPUT.

APPEND War ein File im Speicher, dann wird der Current File Entry Pointer gleich dem End-of-File Pointer, ansonsten werden beide genullt.

Seltsam bei der ganzen Sache ist eine Abfrage bei APPEND auf das FIXED-Format hin. Wollen Sie eine FIXED-Datei mit APPEND eröffnen, so bekommen Sie nur 'Bad Attribute' als Errormeldung. Das Dumme dabei ist, daß es keinen vernünftigen Grund dafür gibt, die APPEND-Funktion nur für VARIABLE-Dateien zuzulassen.

```
G6229  ST    §>834A,§>834B   Statusbyte erneut kopieren
G622C  AND   §>834A,>06      Bits IN/OUT/UPD./APP. abspalten
G622F  CEQ   §>834A,>06      APPEND ?
G6232  BR    GROM§>6244      Nein: Sprung

G6234  CLOG  §>834B,>10      FIXED ?
G6237  BS    GROM§>61B8      Ja: 'Bad Attribute'
G6239  DST   §>0006(§>835A),  Current File Entry Pointer wird gleich
             §>0004(§>835A)   dem End-of-File Pointer
G6242  BR    GROM§>6249      weiter

G6244  DCLR  §>0006(§>835A)      Current File Entry Pointer löschen
G6249  DCLR  VDP§>FFFC(§>8358)   Satznummer im PAB löschen
G624E  CALL  GROM§>0012          GSR-Return to Basic
```

Nun zu CLOSE. Da ist nicht viel zu sagen, da CLOSE nur aus einer einzigen Anweisung besteht:

G6251 BR GROM§>61DE GSR-Return to Basic.

Der Grund ist einleuchtend, wenn man weiß, daß alle Pointer bei jedem Lese- und Schreibzugriff aktualisiert werden. Wird die Datei geschlossen, so muß an ihr nichts mehr verändert werden. Im Gegensatz dazu steht z.B. der Diskcontroller, der die Pointer für den Katalogsektor auf der Diskette während sämtlicher Dateizugriffe im RAM behält und erst bei einem CLOSE wieder aktualisiert zurückschreibt, um den Tonkopf nicht ständig hin- und herfahren zu müssen.

Die letzte Routine für heute ist WRITE, die einen Datensatz in die Datei schreibt, wenn z.B. in Basic ein PRINT # ausgeführt werden muß. WRITE ist im Grunde zweimal vorhanden, einmal für FIXED-Dateien und einmal für VARIABLE-Dateien. Lediglich die MOVE-Anweisung, die die Daten in das Speichersegment schreibt, ist beiden gemeinsam. Außerdem muß natürlich ein Error ('Illegal Operation') ausgegeben werden, wenn in eine Datei geschrieben werden soll, die mit INPUT eröffnet wurde.

VARIABLE Die Routine muß prüfen, ob das Bereichsende erreicht ist (wenn ja, dann 'Buffer ful') und den Current File Entry Pointer und den End-of-File Pointer korrigieren.

FIXED Die Satznummer (REC-Eintrag) wird mit der Satzlänge multipliziert. Die Segmentstartadresse dazu addiert ergibt die Startadresse für den Datensatz, wobei noch auf Bereichsüberschreitungen hin geprüft wird (Error 'Buffer full'). Der Current

File Entry Pointer und der End-of-File Pointer werden korrigiert und die Satznummer im PAB im VDP-RAM um 1 erhöht, damit auch sequentielle Zugriffe ohne Angabe einer REC-Nummer möglich sind.

```
G62C2  ST    §>834A,§>834B  PAB-Status kopieren
G62C5  AND   §>834A,>06     Bits IN/OUT/UPD./APP. abspalten
G62C8  CEQ   §>834A,>04     INPUT ?
G62CB  BS    GROM§>61D8     Ja: 'Illegal Operation'

G62CD  CLOG  §>834B,>10     FIXED ?
G62D0  BS    GROM§>6304     Ja: Sprung

Output, Update oder Append mit VARIABLE
G62D2  DST   §>834A,§>0006(§>835A)  Current File Entry Pointer
G62D8  DST   §>8350,§>834A           "     "     "     "
G62DB  DINC  §>834A         Längenbyte berücksichtigen
G62DD  CLR   §>834E         Hi-Byte löschen
G62DF  DADD  §>834A,§>834E  Zur tatsächlichen Satzlänge den Current
                            File Entry Pointer plus 1 addieren
G62E2  DCH   §>834A,§>835C  Länger als der Bereich ?
G62E5  BS    GROM§>61C0     Ja: 'Buffer full'
G62E7  DST   §>0006(§>835A),§>834A  Neuer Entry Point
G62ED  DST   §>0004(§>835A),§>834A  Neuer End-of-File Pointer
G62F3  DADD  §>8350,§>835A  Relativer Pointer plus Bereichsstartadr.
G62F6  DADD  §>8350,>0008   Plus 8 wegen dem Segmentsheader
G62FA  ST    §>0000(§>8350),§>834F Längenbyte schreiben
G6300  DINC  §>8350         Adresse plus 1
G6302  BR    GROM§>6340     Sprung, um den Datensatz zu schreiben

Output, Update oder Append mit FIXED
G6304  ST    §>834F,§>834E   Max. Länge => Tatsächliche Länge
G6307  CLR   §>834E          Hi-Byte löschen
G6309  AND   §>8350,>7F      MSB der Satznummer löschen
G630C  DST   §>834A,§>8350   Satznummer kopieren
G630F  DMUL  §>8350,§>834E   mit Satzlänge multiplizieren
G6312  DCZ   §>8350          Multiplikationsergebnis größer >FFFF ?
G6314  BR    GROM§>61C0      Ja: 'Buffer full'
G6316  DST   §>8350,§>8352   Adresse kopieren
G6319  DADD  §>8352,§>834E   Satzlänge addieren
G631C  CARRY                 Ergebnis größer >FFFF ?
G631D  BS    GROM§>61C0      Ja: 'Buffer full'
G631F  DCH   §>8352,§>835C   Bereichsende überschritten ?
G6322  BS    GROM§>61C0      Ja: 'Buffer full'
G6324  DINC  §>834A          Satznummer plus 1
G6326  DCH   §>834A,§>0004(§>835A)  größer als End-of-File Pointer ?
G632C  BR    GROM§>6334      Nein: Sprung
G632E  DST   §>0004(§>835A),§>834A  Ja: End-of-File Pointer aktualisieren
G6334  DADD  §>8350,>0008
G6338  DADD  §>8350,§>835A   ergibt die Startadresse des Eintrages
G633B  DINC  VDP§>FFFC(§>8358)  REC-Nummer im PAB plus 1

Datensatz ins File schreiben
G6340  CZ    §>834F          Länge des Datensatzes null ?
G6342  BS    GROM§>834C      Ja: Es muß nichts geschrieben werden
G6344  MOVE  §>834E TO §>0000(§>8350)  Datensatz in das RAM-Segment
                  FROM VDP*>834C        schreiben
G634C  BR    GROM§>61DE      GSR-Return to Basic
```

Eine Anmerkung noch zu G6309: Laut dem Handbuch des TI ist die höchstmögliche REC-Nr. 32757. Höhere REC-Nummern werden nicht akzeptiert bzw. es wird einfach das MSB gelöscht.

Sven Dyroff

TI BÖRSE

TI99/4A-Komplettanlage zu verkaufen. Mit GRAM-Modul (Supermodul!!!). Möglichst kompl. Tel. 02162/56520

Wer baut mir Eprom-Programmiergerät nach W. Jüngst (Heft 6/85) od. nach Heiner Martin (Heft 7/85) f. TI99/4A? Abwicklung über BRD-Verwandt. Zuschriften an Dr. Arnim Tölke, Postf. 11, DDR-1272 Neuenhagen

---

TI-Anfänger! Rufen Sie mich an! Ich verk. günstig bzw. verschenke meine restl. Hard- u. Software. Billiger geht es nicht. F. Meilinger, Fr.-Winter-Str. 43, 4750 Unna, Tel. 02303/60360 nach 17 h

---

Verk.: TI99 m. P-Box, Controller, Disk., 32K-RAM, Speechsynth., Monitor, Ed./Ass., Ex.-Basic, Mini Mem, Eprommer, Mengen an Lit., kompl. 1.000,– DM, möglichst an Selbstabholer (Raum Bochum), evtl. einzeln. K. Kutowski, Tel. 02325/46469

1.) TI99/4A, X-Basic, Speech-Synth., 2 Module, Software, Rec.-Kabel, Joysticks 270,– DM. 2.) TI99/4A, 3 Module, Softw., Rec.-Kabel, div. Lit. 200,– DM. Christian Großmann, Tel. 07024/3104

Disk.-Laufwerk DS DD Slimline f. TI zu verk. Einbau v. 2 dieser Drives in die Box ist möglich. Preis je DM 170,– Näheres: Tel. 06257/83247

---

* Jetzt schlägt's 13! * Zu verkaufen TI-Eprom-Prog.-Gerät m. Modul-Softw. integriert (auch als getrenntes Einschubmodul zu verwenden), dazu 28V Programmier-Netzgerät f. sage u. schreibe 100,– DM. Tel. 07026/5885 abends

---

Verk.: TI99/4A-Konsole, div. Zubehör, z.B. Speech, Mod.-Expand., P-Box (leer), div. Module: XBasic, XB 2, Mini Mem, E/A, Writer, Multipl., Statistik, Logo, div. Spiele, viel Disk.-Softw., Bücher, Zeitschr., Peter Werner, Sandweg 13, 646 Gelnhausen 2, 06051/68700

Verkaufe: 32K, RS232/V24, XB, MM, SS, TI, Box u. LW, 2 D-Boxen, J-Adapter, Lit., GP-500/A, Cen.-Kabel, Mon., Kbel f. 2 LW, def. TI, u.v.m.! VB 2.200,– DM, tausche auch gg. Amiga 500 u. Mon.! Jörg Klass, Tel. 02562/1362

Kompl. TI-Anlage, auch eigene Erweiterungen, preiswert zu verkaufen. L. Kaschke, Wederstr. 59, Berlin 47, Tel. 030/6262950

Suche Ext.-Basic u. evtl. 32K-Erw. f. Direktanschluß an Computer (auch Selbstbau), Tel. 02601/3378

Verk. Mechatronic K-RAM-Disk m. 32K-Erw., Centr.-Anschluß, eig. Stromvers., seitl. anzustecken, durchgef. Bus, läuft o. Box. Preis VB 200,– DM. Tel. 07191/58233 abends

---

Verkaufe: TI99/4A 120,– DM; Kass.-Kabel 20,– DM. Mod.: Defender 20,– DM; Othello 15,– DM; PacMan 20,– DM; Connect Four 15,– DM; Beg. Grammar 10,– DM; Hen House 15,– DM; Rabbit Trail 20,– DM; Alligator Mix u. Alien Addition je 15,– DM. Suche 32K intern u. Rs232 intern. Tel. 08161/65549, Carsten

---

Kaufe TS-Disk.-Triebwerk sowie andere Soft- u. Hardware. Angebote an: Heinz-Jürgen Fischer, Berg.-Gladbacher Str. 800, 5000 Köln 80

Computerfreund aus der DDR sucht f. TI99/4A möglichst kostenlos Ext.-Basic-Modul, Schach, Spielmodule sowie Kontakte zu TI-Clubs. Angebote über Harald Pästges, Grebbener Str. 59, 5138 Heinsberg-Grebben, Telefon 02452/65861

Wer baut mir Achtkanal-Digital-Wandler nach Reiner Ziegler (TI REVUE 8/87) möglichst m. eingebauter Echtzeituhr f. TI99/4A? Od. Achtkanal Ein- u. Ausgabe nach Reiner Ziegler (TI REVUE 2/84) als Modul od. Karte f. P-Box? Abwicklung über BRD-Verwandte. Zuschriften an Dr. Arnim Tölke, Postfach 11, DDR-1272 Neuenhagen

---

Wg. Zeitmangel zu verk.: 2 Konsolen TI99/4A a 90,– DM; orig. Peri-Box m. 32K, TI00 f. 50,– DM; X-Basic 120,– DM; MM 150,– DM; Atronic-Centr.-Kabel 150,– DM; alle TI REVUE 50,– DM; Parsec 30,– DM; 40 Prog.-Kass. 30,– DM; Joystick, Rec.-Kabel u. Rec f. 50,– DM. Alles zusammen: 500,– DM. Tel. 0271/390989

Verkaufe Turbo Pascal, TI E/A, versch. Module. Suche DD/DS Laufwerke u. Contr. DD/DS sowie Kontakt zu TI-User im Raum FR-EM. Alfred Gasperi, Hauptstr. 3, 7808 Waldkirch, Tel. 07681/23757

Suche Asemblerlit. f. TI99/4A. Vorrangig v. simon Koppelmann. Angebote m. Preis an Dr. Arnim Tölke, Postf. 11, DDR-1272 Neuenhagen

---

Verk. Ext. RAM u. Parallel-Schnittst. 200,– DM. Ed./Ass.-Modul m. dt. Handbuch 150,– DM. Tel. 07452/65816

Verk. f. TI99/4A: 32K-Erw., seitl. einzustecken, f. 200,– DM; Schach-Modul 50,– DM; Sprachsynth. 90,– DM. Tel. 09131/44819

Suche dt. Anleitungen f. TI-USCD-PASCAL u. TI-FORTH. Evtl. auch als Kopie. Zuschriften an Dr. Arnim Tölke, Postf. 11, DDR-1272 Neuenhagen

---

Zu verkaufen: TI99/4A Konsole komplett, Modul Ext. Basic, orig. TI, Recorder, Softw., Buch. Alles nur 400,– DM. Tel. nach 19 Uhr 04124/7072

Ex-Basic II Plus m. Manual: 150,– DM; über 300 Prog. auf 55 Kass.: 100,– DM; 10 Module: 100,– DM; Techn.-Data u.a.: 20,– DM; alles zus. f. 300,– DM. Teilzahlung möglich. F. Meilinger, Fr.-Winter-Str. 43, 4750 Unna, Tel. 02303/60360

TI REVUE

Das Magazin für TI 99·4A

Nr. 12

Die DSR-Programmierung

Dem 99/4 A unters Hemd geschaut

Test: Advanced Diagnostics

RGB-Encoder

Tips & Tricks

DAS HEFT II

SERIE & SERVICE

Dem TI unters Hemd geschaut – Teil 2
Interessante Daten zur TI 99/4A-Konsole
Seite 36

Die Programmierung einer DSR – Teil 4
Im vorerst letzten Teil dieser Reihe stellen wir Ihnen die Routinen Input, Restore und EOF vor. Seite 40

Wie wär's mit einem eigenen ROM?
Ein Draht und eine Diode, das ist alles.
Seite 42

TI 99·4A:

TEST & TECHNIK

Dialog
Leserbriefe zum Thema TI 99/4A
Seite 50

Test: Advanced Diagnostics
Mehr als ein Disk-Editor
Seite 47

TIPS & TRICKS

Jetzt wird der TI scharf gemacht
RGB-Encoder Teil 2
Seite 34

Programm zum Ein- und Auslesen von Modulen im Bereich HEX 6000
Seite 43

Programm unter X-BASIC
So lassen sich auch Assembler-Spiele laden.
Seite 45

EINFACH HERAUSTRENNEN

Jetzt wird der TI scharf gemacht

Nachdem in der letzten Ausgabe die Theorie abgehandelt wurde, können Sie nun Ihren Lötkolben vorheizen. Im folgenden Artikel beschreiben wir Aufbau und Abgleich des RGB-Encoders, so daß einem Probelauf nichts mehr im Wege steht.

Nachdem in der letzten TI-REVUE die Schaltung ausführlich besprochen worden ist, kommen wir jetzt zu ihrem Aufbau. Das Platinenlayout beziehungsweise die Platine soll den Nachbau erleichtern. Der notwendige Abgleich wird ebenfalls erklärt.

DER AUFBAU

Der Aufbau des RGB-Encoders ist nicht schwierig. Es sollte darauf geachtet werden, daß auch Drahtbrükken unter den ICs einzulöten sind. Also zuerst die Brücke, dann das IC. Für die Trimmer T7, T9 und T10 sind entweder die normale Ausführung oder die Spindelausführung zu verwenden. Ein schnellerer Abgleich ohne Oszilloskop ist mit der normalen Ausführung zu erreichen. Einer der beiden Synchronausgänge, H oder H+V, wird aktiv, indem eine der vier Lötbrücken mit Lötzinn hergestellt wird.
Für die meisten Kondensatoren sind Lötaugen sowohl im 5- als auch im 7,5-mm-Rastermaß vorhanden.
Die Trimmer für Helligkeit, Kontrast, Farbsättigung und Audio könnten auch durch Potentiometer ersetzt werden, so daß die Handhabung einfacher wird. Vor allem bei einem Gehäuseeinbau hat man so jederzeit Zugriff auf diese Funktionen.

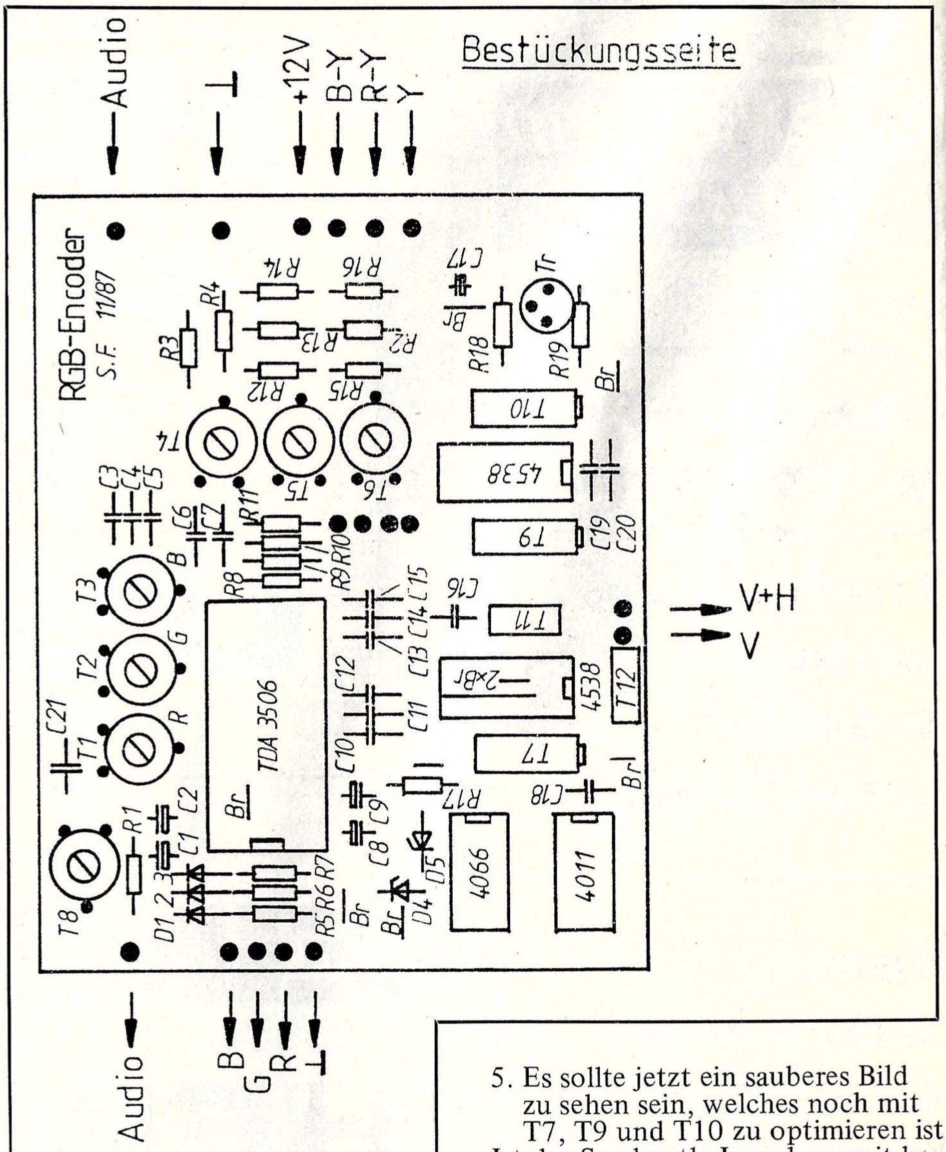
DER ABGLEICH

Die Trimmer T1, T2, T3, T4, T5, T6, T8, T11 und T12 werden in Mittelstellung gebracht. Ist ein Oszilloskop zur Hand, so sollte man den Sandcastle-Impuls folgendermaßen einjustieren:

1. Y-Ablenkung auf 2V/cm X-Ablenkung auf 5us/cm.
2. Mit dem Trimmer T10 die Dauer des 11-V-Impulses auf 4 bis 5 us einstellen.
3. Mit T9 den 4,5-V-Impuls auf ca. 11 us abgleichen.
4. X-Ablenkung auf 1 ms/cm.
5. Mit T7 die Dauer des 2,5-V-Niveaus in der vertikalen Austastlücke auf etwa 1 ms einjustieren. Steht ein Zweikanaloszilloskop zur Verfügung, dann auf den zweiten Eingang das Y-Signal anlegen. In der Zeit, in der kein Bildinhalt vorhanden ist, das heißt, nur Synchronimpulse zu sehen sind, muß das Gleichspannungsniveau 2,5 V betragen.

Wurde alles richtig abgeglichen, sieht der Sandcastle-Impuls so aus:

x-Ablenkung = 5us/cm
X-Ablenkung = 1ms/cm



Ohne Oszilloskop wird die Sache etwas langwieriger, man muß die richtige Einstellung durch Probieren herausfinden:

1. Die Trimmer T7, T9, T10 auf Rechtsanschlag (=maximaler Widerstand) stellen.
2. T7 so weit zurückdrehen, bis die Zeilen verschwunden sind und ein dunkles Bild erscheint. Bis auf die Farbbalken des Computertitelbildes dürfte noch nichts zu sehen sein.
3. Mit T9 durch Linksdrehung den linken dunklen Balken verschwinden lassen. Je mehr gedreht wird, desto heller wird das Fernsehbild. Nur so weit drehen, daß von oben her das Bild nicht verdunkelt wird. Eventuell die Helligkeit und den Kontrast vermindern.
4. T10 nun so weit nach links drehen, bis das Bild dunkel wird, dann wieder ein Stück in die Gegenrichtung drehen.
5. Es sollte jetzt ein sauberes Bild zu sehen sein, welches noch mit T7, T9 und T10 zu optimieren ist.

Ist der Sandcastle-Impuls soweit hergestellt, kann man im Abgleich fortfahren. Als nächstes sollte der Weißabgleich stattfinden. Dazu gibt es zwei Möglichkeiten:

1. Es wird eine Schleife mit CALL SCREEN(16) eingegeben, womit der Bildschirm weiß erscheinen soll. Dann gleicht man mit den Trimmern T1, T2 und T3 nach Gefühl ab.

BAUTEILELISTE

WIDERSTÄNDE:		**KONDENSATOREN:**	
R8, R9, R10, R11	75 Ohm	C19, C20	1,8 nF
R5, R6, R7	180 Ohm	C3, C4, C6, C7	22 nF
R2, R16	1 kOhm	C10, C11, C12	
R17	2,2 kOhm	C13, C14, C15	47 nF
R19	2,7 kOhm	C16, C18, C21, C5	100 nF
R1	10 kOhm		
R3	15 kOhm		
R14	22 kOhm	**TANTALELKOs:**	
R4	39 kOhm	C17	,22 uF
R12	47 kOhm	C1, C8, C9	,33 uF
R18	100 kOhm		
R13	220 kOhm		
R15	330 kOhm	**ELKO:**	
		C2	22 uF
TRIMMER:			
T1, T2, T3,		**TRANSISTOR:**	
T8, T9, T10, T11, T12	10 kOhm	TR	BC 177
T4	50 kOhm		
T5	100 kOhm		
T6	500 kOhm		
T7	2 MOhm		
		ICs:	
DIODEN:		IC1	TDA 3506
D1, D2, D3	1N4148	IC2, IC3	MC 4538
D4	Z 2,7	IC4	MC 4066
D5	Z 6,8	IC5	MC 4011

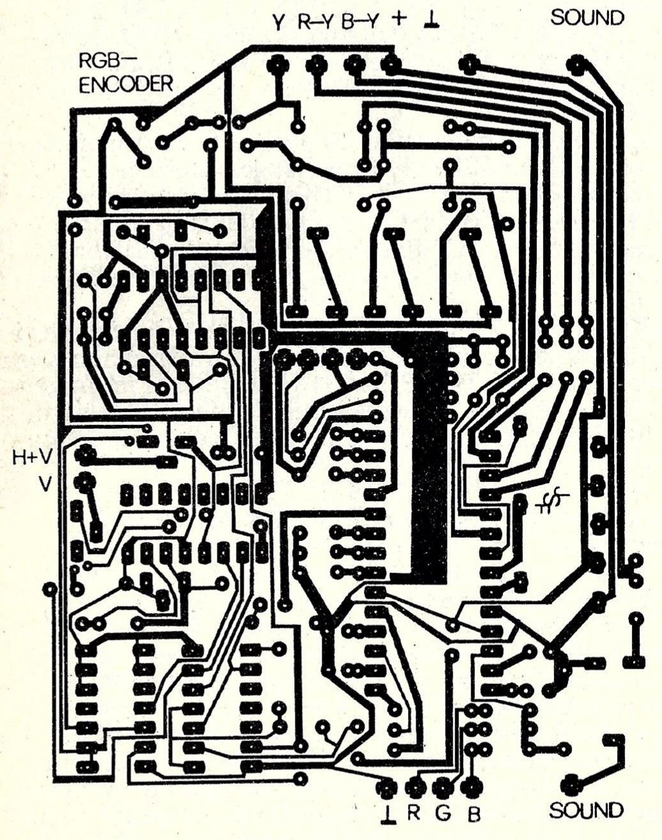
2. Mit den R-G-B-Trimmern wird an Pin 23 etwa 7V, am Pin 22 etwa 4 V und am Pin 21 ebenfalls 4 V eingestellt. Am Prototyp wurden mit diesen Spannungen gute Erfolge erzielt, aber das ist sicher von der Farbempfindlichkeit der Augen eines jeden einzelnen abhängig.

Das gleiche gilt für die Trimmer T4, T5, T6 und T8. T4 ist für die Farbsättigung, T5 für die Helligkeit, T6 für den Kontrast und T8 für die Lautstärke verantwortlich. Sie dienen im Grunde nur der Voreinstellung, da sicher jeder Monitor über diese Einstellmöglichkeiten verfügt.

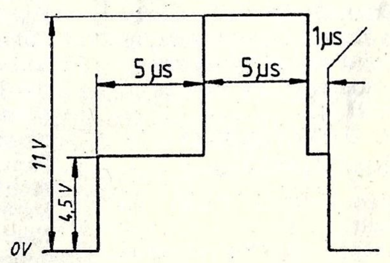
Außerdem ist mit den Trimmern T11 und T12 je nach Bedarf die Synchronimpulsamplitude veränderbar.

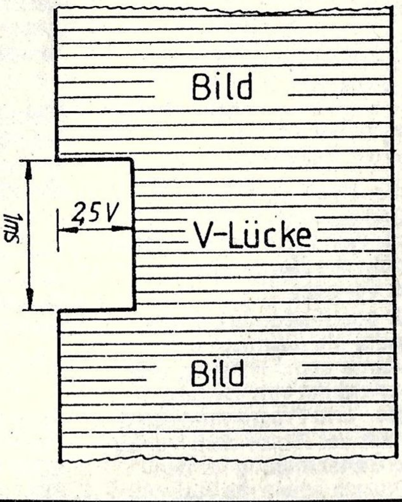
Sollte jemand auf Schwierigkeiten beim Nachbau stoßen, bin ich gerne bereit, weiterzuhelfen. Außerdem besteht die Möglichkeit, den RGB-Encoder fertig aufgebaut und abgeglichen zum Preis von 100 DM, die Platine zum Preis von 20 DM, unter folgender Anschrift zu bestellen:

Siegfried Fränkl
Frauendorferstr. 5
8400 Regensburg
Tel. 0941/96296 □



Das 1:1-Layout der RGB-Encoder-Platine. Dem fortgeschrittenen Bastler kann sie als Vorlage zur eigenen Herstellung der Platine dienen.





Wichtiger Hinweis an alle TI-Abonnenten!

Nachdem die TI-Freaks Sven Dyroff und Jürgen Stelter zu uns gestoßen sind – weitere TI-Experten sind bereits Mitarbeiter –, wird die HCA in den nächsten Ausgabe noch fundiertere Themen zum legendären 99/4A veröffentlichen können. Sollten Sie also Ihr Abo gekündigt haben, blättern Sie doch am Kiosk in den nächsten Monaten mal die HCA durch. Alle Abonnenten – und das ist die überwiegende Mehrheit –, die nicht gekündigt haben, erhalten in den nächsten Tagen die neue Abo-Rechnung, diesmal wieder für zwölf Hefte HCA, bei entsprechender Abbuchungsvollmacht werden wir nach Auslieferung dieses Heftes ebenso die Abbuchung vornehmen. Sie werden es nicht bereuen, der TI-Teil, der in Zukunft allerdings anders aussehen wird, wird noch besser und interessanter. Der Verlag

Die TI-REVUE sucht noch gute Autoren!

IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Tel.: 0 89 / 129 80 11
Telex: 5214428 cav-d

Verantwortlich für den Inhalt:
Lukas Stock (Chefredateur)
Hartmut König (stv. Chefred.)
Redaktion: Thomas Dehn, Sven Dyroff, Christoph Kögler, Hartmut König, Jürgen Puschmann, Uwe Schröder, Jürgen Stelter, Lukas Stock

Geschäftsführer:
Werner E. Seibt

Alle: Postfach 1161 8044 Lohhof.
Anfragen bitte nur schriftlich
Es gilt die Honorarliste des Verlages.
Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung.

Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.

Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

ISSN 0933-3800

© by CA Verlag, Heßstraße 90, 8000 München 40 SPS und Autoren

Dem TI unters Hemd geschaut (II)

Selbst die Entwickler von Texas Instruments waren nicht besser informiert als Sie. Denn auf unserer Exkursion durch die Hardware der TI-Konsole bleibt Ihnen kein Detail verborgen. Lernen Sie die intimen Geheimnisse Ihres Computers kennen.

1.3.2 Speicherzugriffsprotokolle

In der Konsole sind vier verschiedene Speicherzugriffsprotokolle realisiert. Im folgenden werden alle Protokolle erläutert und die davon betroffenen Speicherbereiche genannt.

a) Interner 16 Bit-breiter Zugriff

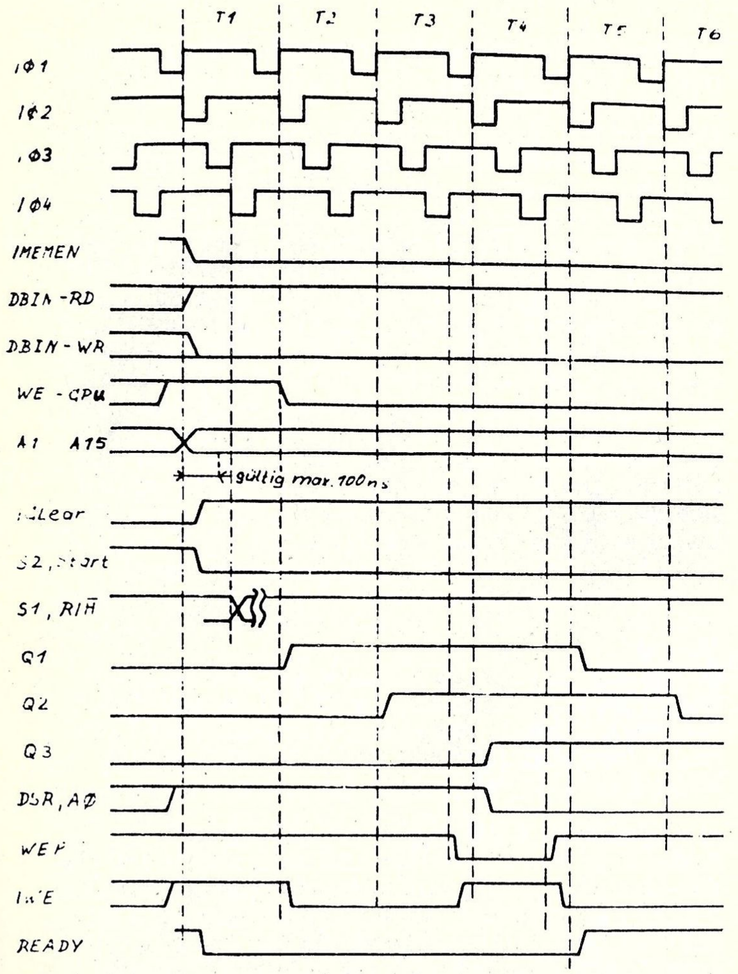
Dieser Speicherzugriff ist der leistungsfähigste und dauert genau zwei Systemtaktperioden. Dieser Zugriff wird auf dem Betriebssystem-ROM im Segment 0 (>0000 ... >1FFF) und auf dem Scratch-Pad-RAM auf den Adressen >8300 ... >83FF realisiert. Dekodiert wird ein solcher Zugriff durch das NAND-Gatter U606.4. Ist einer der Eingänge low-aktiv, so ist der Ausgang high und damit auch der Ausgang des nachgeschalteten OR-Gatters U 605.4, und die später erläuterte Ablaufsteuerung kann nicht starten. Alle Datenbustreiber und -latches bleiben inaktiv, und es werden direkt an den CPU-Datenbus angeschlossene Quellen beziehungsweise Ziele adressiert. Zur Dekodierung werden die Signale /ROMEN und /RAMEN, das mit A10 ... A12 = Low getorte /MBI-Signale, verwendet (Gatter U 507.1, U 507.3, U 507.4). Dabei wird gleichzeitig das /CS-Signal für die CRU-Adressierung des Peripheriebausteins TMS9901 am Ausgang des OR-Gatters U 507.4 erzeugt (/CRU-CS-TMS9901).

Dieser Speicherzugriff kann durch ein extern eingespeistes Ready-Signal low verlängert werden. Der READY-D-Trigger U 607.2 tastet die multiplex genutzte Ready-/Hold-Leitung mit der steigenden Flanke von /CLK2 ab und kann über die Gatter U 603.4 und U 602.2 Wait-Zyklen der CPU erzwingen.

Alle anderen Speicherzugriffe benutzen zur Synchronisation des Ablaufs die Ablaufsteuerung, deren Kern der Schaltkreis U 613 74LS194 bildet. Da im folgenden die Kenntnis dieses Ablaufs vorausgesetzt wird, soll diese Ablaufsteuerung hier genauer beschrieben werden.

Erläuterungen zur Funktionsweise der Ablaufsteuerung

Gestartet wird die Ablaufsteuerung durch jeden Speicherzugriff, der nicht zu der unter a) genannten Art gehört, das heißt, wenn die Eingänge des NAND-Gatters U 606.4 high sind und damit der Ausgang auf low liegt. Über die Verknüpfung mit /MREQ=low an dem OR-Gatter U 605.4 wird dessen Ausgang low-aktiv, und über den Negator U 604.1 wird das Rücksetzsignal an /RD des 74LS194 zurückgenommen. Der Eingang S2 bleibt für die Zeit des Speicherzugriffs auf low, und der Eingang S1 kann die Zustände low (an der Leitung R/H' wurde ein low abgetastet) oder high annehmen. Im Fall von low geht das



Erzeugung des virtuellen Adreßbit ABØ sowie READY-Ablaufsteuerung

Schieberegister wegen S1=S2=low in den Zustand „blockiert" und es reagiert nicht mehr auf den Takteingang, bis die R/H'-Leitung wieder auf high geht und damit S1 auf high.
Für den Fall S1=high und S2=low ist das Schieberegister auf Rechtsschieben programmiert.
Ein über die Ablaufsteuerung synchronisierter Speicherzugriff ist immer (mindestens) sechs Takte lang. In Abbildung 2 ist das Zeitdiagramm der Ablaufsteuerung angegeben.

Takt 1: Das Schieberegister befindet sich im Rücksetzzustand und alle Ausgänge sind auf low, über den Negator U 602.4 wird das virtuelle Adreßbit A0=high erzeugt und ausgegeben. Das bedeutet, es erfolgt ein Zugriff auf einer ungeraden Adresse. Da Q3 des Schieberegisters auf low liegt, ist der Ausgang des NAND-Gatters U 612.2 auf high. Durch Verknüpfung mit dem inaktiven Rücksetzsignal=high am NAND-Gatter U 603.1 wird der Ausgang desselben low-aktiv und damit über die Gatter U 603.4 und U 602.2 der Ready-Eingang der CPU auf low gezogen; sie wird also in den WAIT-Zustand Zustand gebracht.
Handelt es sich um einen Schreibzyklus (DBIN=low), dann wird über das NAND-Gatter U 603.3 durch Verknüpfung mit dem virtuellen Adreßbit A0 der Datentreiber U 616, 74LS244, der den low-Teil des 16-Bit-Datenbusses auf den externen Datenbus ED0 ... ED7 legt, durch das Signal /DOG aktiviert.

Takt 2: Mit der steigenden Flanke von /CLK1 wird A0=high über DSR nach Q1 geschoben. Damit wird das NAND-Gatter U 612.1 schaltfähig und zieht seinen Ausgang auf low, und damit wird wie oben beschrieben die Wait-Erzeugung weiter aufrechterhalten.
Q2 führt immer noch low-Potential. Wenn die CPU einen Schreibzyklus vollführt, geht jetzt die /WE*-Leitung auf low. Q2 setzt immer noch den WE-D-Trigger U 607.2, so daß das Ausgangssignal WEP über das NAND-Gatter U 606.1 einen Schreibimpuls /WE erzeugen kann.
Im Falle eines Schreibzyklus bleibt der Datentreiber U 616, 74LS244 weiter aktiv.

Takt 3: Mit der steigenden Flanke von /CKL1 wird das high von Q1 nach Q2 geschoben, über DSR wird immer noch high nach Q1 geliefert. Damit sind jetzt Q1 und Q2 auf high, Q3 (und Q4) auf low. Es verhält sich alles so wie im Takt 2 mit der Ausnahme, daß inzwischen das Setzsignal für den WE-D-Trigger zurückgenommen wurde. Damit wird der Trigger mit der steigenden Flanke von /CLK4 das low von Q3 übernehmen und im Falle eines laufenden Schreibzyklus über das NAND-Gatter U 606.1 das /WE-Signal beenden, während das der CPU weiterhin bestehen bleibt.

Damit ist der erste Teil des in zwei Acht-Bit-Zugriffe serialisierten 16-Bit-Speicherzugriffs gelaufen, es wird immer zuerst auf die ungerade Adresse zugegriffen (A0=1), also auf die low-Bank.

Takt 4: Mit der steigenden Flanke von /CLK1 wird das high von Q2 nach Q3 übernommen und über DSR immer noch high nach Q1. Damit sind inzwischen die Ausgänge Q1 ... Q3 auf high, und Q3 liefert über Negator U 602.4 die Adresse A0=0, womit der Zugriff auf die High-Bank, die gerade Adresse, eingeleitet wird.
Zu diesem Zeitpunkt sind aber auch die Daten eines eventuell gelaufenen Lesezyklus auf der Low-Bank gültig. Sie werden mit dem Übergang A0 high –> low im Datenlatch U 615, 74LS373 zwischengespeichert, damit die CPU sie später parallel mit den noch zu holenden Daten der High-Bank lesen kann.
Durch den Zustand A0=low wird über das NAND-Gatter U 603.3 der Datentreiber 74LS 244 der Low-Bank abgeschaltet. Da Q1 noch auf high ist, bleibt auch die WAIT-Erzeugung aktiviert, Ready auf low. Im Falle eines Lesezyklus werden die auf der Low-Bank gelatchten Daten durch Verknüpfung von /A0 und DBIN' am NAND-Gatter U 603.2 und Erzeugung des Signals /DING zum Low-Teil des Datenbusses der CPU zum Lesen getrieben.
Jetzt wird der bidirektionale Datentreiber U 614, 74LS245 der High-Bank durch Verknüpfung der Signale /A0 und /VDPREAD am NAND-Gatter U 606.2 und Erzeugung des Signals /DIOG aktiviert, die Richtung des Transfers wird durch /DBIN gesteuert. Im Falle eines Schreibzyklus werden jetzt die Daten des High-Teils des CPU-Datenbusses auf den externen Datenbus gelegt beziehungsweise die gelesenen Daten zur CPU übertragen.
Am Ende von Takt 4 wird durch die steigende Flanke von /CLK4 das high von Q3 durch

den WE-D-Trigger übernommen, und im Falle eines Schreibzyklus wird erneut das Signal /WE über das NAND-Gatter U 606.1 erzeugt.

Takt 5: Mit der steigenden Flanke von /CLK1 wird jetzt über DSR ein low nach Q1 geschoben, Q2 und Q3 bleiben noch auf High-Potential. Wegen Q1=low wird der Ausgang des Gatters U 612.1 auf high gehen. Durch Verknüpfung mit dem high von Q3 über das NAND-Gatter U 612.2 wird dessen Ausgang auf low gehen und damit den Ausgang des NAND-Gatters U 603.1 auf high zwingen. Liegt nun keine externe WAIT-Anforderung über den READY-D-Trigger vor, sind beide Eingänge des NAND-Gatters U 603.4 auf high. Damit wird der Eingang READY der CPU auf high gezogen und es werden keine weiteren WAIT-Zyklen mehr erzeugt werden. Ansonsten verhält sich alles wie im Takt 4.

Takt 6: Mit der steigenden Flanke von /CKL1 wird das low von Q1 nach Q2 übernommen und weiterhin low über DSR nach Q1 nachgeschoben. Jetzt sind Q1 und Q2 auf low und Q3 auf high. Dadurch bleibt die WAIT-Generierung abgeschaltet, sofern nicht ein externes READY=low vorliegt. Q2 liegt auf low und setzt damit zwangsweise den WE-D-Trigger, so daß der eventuell erzeugte /WE-Impuls erst mit dem Verschwinden des /WE*-Signals der CPU wegfällt. Ein eventueller Lesezyklus findet am Ende von Takt 6 seinen Abschluß, wo die CPU die Daten mit der Breite von 16 Bit übernimmt.

Die nächste steigende Flanke von /CLK1 beziehungsweise das Verschwinden des Signals /MREQ führen zum Rücksetzzustand des Schieberegisters und erzeugen somit wieder das virtuelle Adreßbit A0=1. Damit werden der bidirektionale Treiber 74LS245 über das NAND-Gatter U 606.2 und das Datenlatch 74LS373 über das NAND-Gatter U 603.2 abgeschaltet. Jetzt ist der Speicherzugriff beendet. Er setzte sich aus T1 des normalen Speicherzugriffs (Takt 1), vier WAIT-Takten (Takte 2 bis 5) und T2 des normalen Speicherzugriffs zusammen. Über die R/W'-Leitung eingeblendete WAIT-Takte verlängern das Speicherzugriffsprotokoll proportional. Alle nicht „internen" 16 Bit breiten Speicherzugriffe laufen unter Synchronisation der Ablaufsteuerung und sind daher zwangsweise mindestens sechs T-Zyklen lang. Erzeugen die angesprochenen Quellen oder Senken ein /READY-Signal, dann wird die Ablaufsteuerung so lange blokkiert, bis dieses Signal verschwindet.

b) Zugriff auf den Videoprozessor

Schreib- und Lesezugriffe auf den Videoprozessor werden mit der IS U505 (74LS138) durch Erzeugung der Signale /VDPREAD und /VDPW selektiert. Da das Lesen oder Schreiben aus der Adresse abgeleitet wird (>8800 READ, >8C00 WRITE), ist noch eine Sicherung gegen ungewollte Fehladressierung vorgenommen worden. Wenn es sich um einen VDP-READ-Zyklus handelt, dann ist das Statussignal DBIN' high-aktiv. Gleichzeitig darf das Adreßbit A10 nicht high-aktiv sein, da es sich sonst auf einen Speicherplatz zwischen >8C00 und >8FFF bezieht. Diese Verriegelung wird mit dem NAND-Gatter U 506.2 vorgenommen, das im Fehlerfall den 1 aus 8-Dekoder gar nicht selektiert. Ein erzeugtes Signal /VDPW wird zusätzlich mit dem Signal /WE getort (OR-Gatter U 507.2), damit ein Lesezyklus auf einer Schreibadresse nicht versehentlich zu Fehlern führt.

Der Videoprozessor ist (wie alle memory-mapped Peripherie) an den High-Teil des 16-Bit-Datenbusses der CPU angeschlossen. Im Gegensatz zu den anderen I/O-Geräten ist der Videoprozessor aber intern, das heißt, direkt mit dem Datenbus ID8 ... ID15 verbunden. Da der Speicherzugriff über die oben erläuterte Ablaufsteuerung synchronisiert wird, muß dafür Sorge getraten werden, daß sich der Videoprozessor und der bidirektionale Treiber 74LS245 nicht gegenseitig stören. Bei einem Schreibzyklus ist das ohnehin nicht der Fall, die Daten werden dem Videoprozessor und dem externen Datenbus ED0 ... ED7 parallel angeboten. Videoprozessorschreibzyklen sind also auf dem externen Datenbus parallel existent. Stören würden sich die beiden Geräte aber im Fall eines Lesezyklus. Daher wird das Freigabesignal für den 74LS245 /DIOG am NAND-Gatter U 606.2 mit dem Selectsignal /VDPREAD getort, so daß der Treiber nur aktiv wird, wenn es sich nicht um einen VDP-READ-Zyklus handelt.

Da der Zugriff über die Ablaufsteuerung synchronisiert wird, erfolgt erst ein Zugriff auf der ungeraden Adresse mit A0=1. Der geht jedoch immer ins Leere, da der Dekoder U505 erst mit dem Signal A0=0 selektiert wird.

c) Allgemeiner 8-Bit-Zugriff auf I/O-Geräte

Außer dem Videoprozessor sind noch andere I/O-Geräte (memory mapped im Speichersegment >8000 ... >9FFF) ansprechbar. Allen ist gemeinsam, daß sie auf dem High-Teil des Datenbusses an den Prozessor angekoppelt sind, sich aber am externen Datenbus ED0 ... ED7 befinden. Die Selektion erfolgt über das Adreßbit A0=0. Wie schon beim Videoprozessorzugriff geht hier der Teil des Zugriffs mit A0=1 ins Leere, da schon der Dekoder U505 nicht selektiert wird. Der Datentransfer erfolgt während der Takte 4 ... 6 der Ablaufsteuerung über den bidirektionalen Treiber. Von dieser Zugriffsart machen der Soundgenerator, das SPEECH-Modul und die „G"-Speicherebene Gebrauch. Unter Umständen dauern die Zugriffe länger als die sechs Takte der Ablaufsteuerung, da zum Beispiel der Soundgenerator die R/H'-Leitung auf low zieht und für diese Zeit zusätzliche WAIT-Zyklen erzeugt werden. Das Stoppen der Ablaufsteuerung ist notwendig, damit der eigentliche Datentransfer nach Rücknahme des READY=low in die Takte 4 bis 6 der Ablaufsteuerung fallen. Nur dort wird der bidirektionale Treiber zum High-Teil des CPU-Datenbusses selektiert.

d) Zwei mal 8-Bit-Zugriff auf eine Speichererweiterung

Wird weder der interne Speicherzugriff (/ROMEN, /RAMEN) noch der 8-Bit-Zugriff auf I/O-Geräte im Segment /MBI (ausgenommen Scratch-Pad-RAM) selektiert, dann handelt es sich um einen zwei 8-Bit-Zugriffe serialisierten 16-Bit-Zugriff auf externen Speicher am Modulport (>6000 ... >7FFF) oder am I/O-Port (>2000 ... >5FFF, >A000 ... >FFFF). Dieser Zugriff läuft exakt so ab, wie bei der Ablaufsteuerung beschrieben. Zusätzliche WAIT-Zyklen sind durch low-Steuern der R/H'-Leitung möglich, werden aber im allgemeinen durch die Baugruppen (Module, DSR-Routinen, Speichererweiterung) nicht benutzt.

Die grundsätzliche Verwendung der Ablaufsteuerung für die Synchronisation allgemeiner Speicherzugriffe ist vor allem an der häufig kritisierten Trägheit des TI 99/4A schuld. Im Abschnitt Evolution /Revolution werden dazu Alternativen aufgezeigt, die den TI 99/4A das Laufen lehren.

2. DIE COMMUNICATION REGISTER UNIT CRU

Der Mikroprozessor TMS 9900 verfügt nicht über einen parallel adressierbaren I/O-Bereich, wie das allgemein bei anderen Prozessoren üblich ist. Statt dessen verfügt er über je einen vier KBit großen Eingabe- und Ausgabebereich, bei dem jedes Bit einzeln adressiert wird. Dazu dienen die Adressen A1 ... A12. Das virtuelle Adreßbit der TI 99/4A-Konsole spielt dabei natürlich keine Rolle. Der Datentransfer läuft über je einen speziellen Eingang (CRUIN) und einen Ausgang (CRUOUT). Zur Unterscheidung von einem herkömmlichen Speicherzugriff wird statt des Signals /MREQ das Signal CRUCLK aktiviert. Ein CRU-Zugriff ist immer genau zwei T-Zyklen lang.
Der Mikroprozessor verwendet die CRU-Adressen auch noch zu anderen Zwecken als zur Adressierung der I/O-Peripherie. Diese liegt nur vor, wenn während des Zugriffs die oberen drei Adreßbit A13 ... A15 =0 sind. Sonst handelt es sich um Ausführungen von speziellen, im Befehlssatz des TMS 9900 enthaltenen, Befehlen. Sie können nützlich sein, um Hardware-Komponenten einer Rechnerkonfiguration zu bedienen.
Näheres zu diesen Befehlen findet man in jeder Beschreibung des Befehlssatzes des TMS 9900. Wichtig für den Betrieb der Konsole des TI 99/4A ist, daß man auf die Ausdekodierung dieser oberen drei Adreßbit verzichtet hat. Das hat zur Folge, daß fünf spezielle Befehle des Befehlssatzes des TMS 9900 in Maschinenprogrammen nicht verwendet werden dürfen, unter anderem auch der HALT-Befehl (IDLE). Geschieht dies trotzdem, so werden die CRU-Operationen fälschlicherweise als I/O-Operationen interpretiert. Dadurch kann die Initialisierung des internen Peripherieschaltkreises TMS 9901 geändert oder irgendein über das I/O-Port angeschlossenes I/O-Gerät ungewollt angesprochen werden. Im Abschnitt Evolution wird darauf eingegangen, wie dieser Nachteil aufgehoben werden kann.
In der TI 99/4A-Konsole wird die Leitung für das virtuelle Adreßbit A0 multiplex mit der Leitung CRUOUT genutzt. Dazu dient folgende logische Verknüpfung:
Das virtuelle Adreßbit A0 ist im Ruhezustand immer auf 1 gestellt und damit high-aktiv. Es wird an dem NAND-Gatter U 606.3 mit dem Signal OUTC verknüpft und anschließend mit dem als Negator arbeitenden NAND-Gatter U 612.4 invertiert. Das mit dem OR-Gatter U 605.3 erzeugte Signal OUTC ist nur dann high-aktiv, wenn entweder das Signal CRUOUT oder das über den Negator U 604.2 invertierte Signal /MREQ aktiv ist.

Wenn /MREQ aktiv ist, handelt es sich um einen echten Speicherzugriff, und das Signal CRUOUT darf nicht von Belang sein; OUTC ist high-aktiv und das Signal A0/CRUOUT ist nur abhängig vom Zustand der Leitung A0. Ist daagegen /MREQ inaktiv, dann wird OUTC nur high-aktiv, wenn auch CRUOUT high-aktiv ist. Da im Falle einer CRU-Operation das virtuelle Adreßbit A0 in Ruhelage und damit high-aktiv ist, repräsentiert die Verknüpfung über die NAND-Gatter U 606.3 und U 612.4 genau den Zustand von CRUOUT.
Innerhalb der Konsole ist nur ein Schaltkreis über die CRU adressiert: der TMS 9901, der zur Tastaturverwaltung, zum Joystickanschluß und zur Interruptverwaltung herangezogen wird. Selektiert wird dieser Schaltkreis mit der Leitung CRU-CS-9901, die von den OR-Gattern U 507.1 und U 507.4 angesteuert wird und die Adreßleitungen A10 ... A12 ODER-verknüpft. Dieser Schaltkreis wird also genau dann adressiert, wenn alle drei Adreßleitungen auf 0 stehen. Das entspricht einem CRU-Adreßbereich von >0000 bis >03FE.
Die Aussage in der Beschreibung des TI 99/4A, die CRU-Adressen unterhalb >1000 seien intern vergeben, ist also unrichtig. Nur der genannte Bereich ist intern vergeben, die Adressen >0400 ... >0FFE stehen dem Anwender zusätzlich zur Verfügung. Dort kann man sinnvollerweise solche nützliche Schaltkreise wie 74LS251 oder 74LS259 adressieren und so das Anwendungsgebiet der CRU erweitern.

3. DER INTERRUPTKONTROLLER TMS 9901

Wie im Kapitel CRU schon erwähnt, ist der Controller im CRU-Adreßbereich >0000 ... >03FE angeordnet. Da er selbst nur 32 CRU-Bit beansprucht (A1 ... A5), ist dieser Bereich nicht voll ausdekodiert, der Adreßbereich spiegelt sich 16mal in dem genannten Segment.
Zur Abfrage der Tastatur und des Joysticks werden über den Schaltkreis U 300, 74LS155 sechs Spalten zur Tastaturabfrage und zwei Spalten zur Joystickabfrage über die Leitungen KCD0, KCD1 und KDSL selektiert, das heißt, nach low gezogen. Die Antwort der Tastaturmatrix und der Joystickkontakte wird multiplex über die Leitungen KEY1 ... KEY7 sowie KEY10 und KEY11 abgefragt. Alle Leitungen werden, über Tiefpässe, funkentstört aus der geschirmten Konsole herausgeführt.
Die beiden Joysticks werden über die Leitungen JOY2 beziehungsweise JOY7 selektiert, die über je einen Emitterfolger aus den Leitungen JST2 und JST7 gespeist werden. Die Leitungen JOY2 und JOY7 führen daher im Ruhezustand High-Potential. Das ist jedoch durch die Tiefpaßwiderstände so hochohmig, daß die Tastaturselektleitungen KEY8 und KEY9 beziehungsweise KEY12 ... KEY15 sich durchsetzen können, auch wenn am Joystick gespielt wird.
Zur Steuerung der beiden anschließbaren Kassettenlaufwerke werden die Leitungen CAS1 und CAS2, über Optokoppler galvanisch entkoppelt, am Steckverbinder als elektronischer Schaltkontakt bereitgestellt.
Die von einem Kassettenrekorder kommenden Daten werden symmetrisch eingekoppelt, mit einem OPV verstärkt und mit dem nachfolgenden Trigger auf Rechteckformat gebracht. Dann werden sie über die Leitung AUDIN zum TMS 9901 übertragen.
Gleichzeitig wird das Signal MAGIN über einen Entkoppelwiderstand von 10 kOhm zu dem Schalttransistor TIS 92 geleitet, der über das Signal /AUDEN das Signal nach Masse kurzschließen kann. Damit kann es, ohne das Lesen von Kassette zu stören, nicht mehr über den folgenden 10 kOhm-Widerstand zum Mischpunkt am Soundgenerator gelangen. So wird es möglich, von Kassette Daten zu lesen, ohne daß sie im Tonkanal beim nachfolgenden Verstärker landen. Weiterhin übernimmt der TMS 9901 die Kontrolle der externen Interrupts sowie die das Video-Control-

lers. Grundsätzlich verwendet die Konsole nicht die Möglichkeit des gestuft vektorisierten Interrupts, sondern es wird immer der Interruptlevel 1 (fest verdrahtet) zur CPU übertragen. Die Lokalisation der Interruptquelle wird über Softwareorganisation erledigt. Das ändert jedoch nichts an der Tatsache, daß der externe Interrupt gegenüber dem Bildschirminterrupt priorisiert ist; lediglich die Vektorinformation wird nicht übertragen.

6. DIE BUSSYSTEME AN DER TI 99/4A-KONSOLE

Alle Adreßleitungen werden vor Ausgabe am I/O-Port über die Schaltkreise 74LS244 sowie 74LS367 getrieben, jedoch nicht am Modulport. Diese Tatsache hat schon oft zu Störungen und Defekten an der Konsole geführt. Auch die Signale /CLK3, /CRUCLK und /WE für beide Ports sind getrieben. Ebenfalls der externe Datenbus ED0 ... ED7 (wie im Rahmen der Ablaufsteuerung beschrieben), so daß zumindest am I/O-Port nach außen keine direkten CPU-Anschlüsse existieren, von einigen Steueranschlüssen abgesehen.

Der externe Datenbus wird über acht Widerstände von 2,2 kOhm bei jedem Lesezyklus über den Negator U 604.4 und den nachfolgenden Schalttransistor TIS 92 nach Masse gezogen, so daß ein Zugriff auf eine nicht angeschlossene Quelle immer die Information >00 ergibt. Handelt es sich jedoch um einen Schreibzyklus, so übernehmen diese Widerstände die Funktion eines „Pull Up" gegen +5 Volt, die über die Diodenkombination erzeugt werden. Das dient offensichtlich der Anpassung an die angeschlossenen GROM's. Ebenfalls dazu wird für die GROM's über die Spannung –U5, einem Widerstand von 270 Ohm und einer Klemmdiode nach Masse eine separate GROM-Masse, bezeichnet als –U0, erzeugt.
Die Statusinformationen HOLDA beziehungsweise IAQ werden multiplex auf einer gemeinsamen Leitung HOLDA/IAQ ausgegeben. Um welche der beiden Informationen es sich handelt, kann extern durch Verknüpfung mit /HOLD separiert werden, da sich das Busanforderungssignal in jedem Fall durchsetzen wird und ein erneuter Befehlslesezyklus bei aktiviertem /HOLD nicht möglich (Hold ist aber meistens gar nicht angeschlossen). *Heinrich Cronjaeger* □

Die Programmierung einer DSR - Teil IV

Nachdem im letzten Bericht die Routinen OPEN, CLOSE und PRINT besprochen wurden, möchte ich in diesem abschließenden Teil die übrigen Routinen, nämlich INPUT, RESTORE und EOF, vorstellen.
Die INPUT-Routine fragt zunächst ab, unter welchen Attributen die Datei eröffnet wurde. Lauten diese auf OUTPUT oder APPEND, so kann die Datei nicht gelesen werden und es gibt einen Error "Illegal Operation". Ansonsten wird auf FIXED beziehungsweise VARIABLE geprüft und dementsprechend verzweigt.

Start der Routine READ:

```
G6253 CLOG  §>834B,>02  OUTPUT o. APPEND?
G6256 BR    GROM§>61D8 Ja: 'Illegal Operation'
G6258 CLOG  §>834B,>10  FIXED?
G625B BS    GROM§>6287 Ja: Sprung
```

Bei einer VARIABLE-Datei muß der Computer zuerst abfragen, ob das Dateiende bereits erreicht ist. Wenn ja, dann gibt er den Error "Read past EOF" aus:

INPUT oder UPDATE mit VARIABLE:

```
G625D DCHE  §>0006(§>835A),  Current-File-
            §>0004(§>835A)   Entry-Point grö-
                             ßer oder gleich
                             dem End-Of-File-
                             Point?
G6266 BS    GROM§>61C8       Ja: "Read past
                             EOF"
```

Ist alles in Ordnung, so holt er sich den Current-File-Entry-Pointer aus den Byte 6 und 7 des Segment-Headers, addiert die Startadresse der Datenfiles hinzu und liest den Längenpointer des entsprechenden Datensatzes:

```
G6268 DST   §>834A,§>0006   Current-File-Entry-
            (§>835A)        Point holen
G626E DADD  §>834A,         plus Startadresse
            §>835A          Speichersegment
G6271 DADD  §>834A,>0008    plus Headerlänge
G6275 ST    §>834F,§>0000   Längenpointer
            (§>834A)        holen
G627B CLR   §>834E          Hi-Byte löschen
```

Diese Längenausgabe muß zu dem Current-File-Entry-Pointer in den Byte 6 und 7 addiert werden, damit dieser, wenn er dann noch in G62A9 um eins hochgezählt wird, auf den nachfolgenden Datensatz zeigt:

```
G627  DINC  §>834A          wegen dem Längen-
G627F DADD  §>0006          byte neuer Current-
            (§>835A),       File-Entry-Point-
            §>834E          Sprung
G6285 BR    GROM§>62A9
```

Bei FIXED-Dateien erfolgt natürlich ebenfalls zuerst die Abfrage, ob das Dateiende bereits erreicht ist, wobei Sie beachten müssen, daß, wie in Teil III bereits aufgeführt, bei FIXED-Routinen diese Pointer REC-Nummern enthalten.

INPUT oder UPDATE mit FIXED:

```
G6287 DCGT  §>0004          End-of-File größer
            (§>835A),       aktuelle REC-Nr.?
            §>8350
G628D BR    GROM§>61C8      Nein: "Read past
                            EOF"
```

Nun muß der TI sich die Startadresse der zu lesenden Datensätze errechnen. Er nimmt dazu die aktuelle REC-Nummer, multipliziert sie mit der Satzlänge und addiert die Startadresse des Datenfiles hinzu:

G628F ST	§>834F,§>0003 (§>835A)	Logical Record Length (Satzlänge)
G6295 CLR	§>834E	Hi-Byte löschen
G6297 DMUL	§>8350,§>834E	REC-Nummer mal Satzlänge
G629A DST	§>834A,§>8352	Ergebnis kopieren
G629D DADD	§>834A,§>835A	plus Startadresse Speichersegment
G62A0 DADD	§>834A,>0008	plus Länge des Headers

Jetzt muß er nur noch die aktuelle REC-Nummer im PAB um eins erhöhen, damit beim nächsten Zugriff nicht wieder der gleiche Datei-Eintrag gelesen wird:

G62A4 DINC	VDP§>FFFC (§>8358)	REC-Nummer im PAB plus eins

Was nun folgt, ist wieder sowohl für FIXED- als auch für VARIABLE-Dateien gültig. Der Current-File-Entry-Pointer wird um eins erhöht, der Datensatz gelesen und die Satzlänge im PAB vermerkt, da diese bei VARIABLE-Dateien naturgemäß bei jedem Eintrag unterschiedlich sein kann.

Satz aus dem File lesen:

G62A9 DINC	§>0006 (§>835A)	Current-File-Entry-Pointer + 1
G62AE CZ	§>834F	Satzlänge gleich Null?
G62B0 BS	GROM§>62BA	Ja: Da ist nichts zum Lesen
G62B2 MOVE	§>834E TO VDP*>834C FROM §>0000 (§>834A)	Satz lesen
G62BA ST	VDP§>FFFB (§>8358), §>834F	Satzlänge im PAB vermerken
G62C0 BR	GROM§>61DE	GSR-Return to Basic

Die Routine RESTORE ist ausgesprochen kurz. Bei VARIABLE-Dateien wird der Current-File-Entry-Pointer einfach gelöscht, wodurch er zum Dateianfang zeigt, wenn die Startadresse des Datenfiles aufaddiert wird. Bei FIXED-Dateien wird in ihn die REC-Nummer geschrieben, die im 7. und 8. Byte des PABs steht, da bei RELATIVE-Dateien auf jeden beliebigen Datensatz zurückgesprungen werden kann. Ist die Datei sequentieller Art, so muß der Benutzer eben Sorge tragen, daß bei einem RESTORE-Aufruf diese beiden Byte gleich Null sind.

Start der Routine RESTORE:

G634E CLOG	§>834B,>10	FIXED?
G6351 BR	GROM§>635E	Nein: Sprung

FIXED-Datei:

G6353 DST	§>0006(§>835A), VDP§>FFFC (§>8358)	Satznummer aus PAB in den Current File-Entry-Pointer
G635C BR	GROM§>61DE	GSR-Return to Basic

VARIABLE-Datei:

G635E DCLR	§>0006 (§>835A)	Current-File-Entry-Pointer löschen
G6363 BR	GROM§>61DE	GSR-Return to Basic

Zum Abschluß noch EOF: Es fragt bekanntlich die charakteristischen Parameter einer Datei ab und setzt dementsprechend Bit im 9. Byte des PAB. Die Routine schreibt diese Bit zunächst alle in >834A und kopiert dieses Byte dann zum Schluß in den PAB. Sie beginnt also damit, daß alle Bit in >834A gelöscht werden und anschließend abfragt, ob denn überhaupt Dateien im Speicher liegen. Wenn nicht, wird das Nullbyte in den PAB geschrieben und zum Basic zurückgekehrt.

Start der Routine EOF:

G63C3 CLR	§>834A	EOF-Byte löschen
G63C5 DCEQ	§>0000 (§>835A), >5AA5	überhaupt Dateien im Speicher?
G63CC BR	GROM§>642B	Nein: Byte in den PAB und Ende

Ist eine Datei vorhanden, so gibt es drei Möglichkeiten: PROGRAM-, FIXED- und VARIABLE-Dateien. Bei PROGRAM-Dateien ist die Abfrage am kürzesten, da nur das 5. Bit gesetzt werden muß:

G63CE ST	§>834B, §>0002 (§>835A)	Dateistatusbyte holen
G63D4 CEQ	§>834B,>FF	PROGRAM-File?
G63D7 BR	GROM§>63DE	Nein: Sprung

Speicher enthält ein PROGRAM-File:

G63D9 ST	§>834A,>08	5. EOF-Bit: PROGRAM-File
G63DC BR	GROM§>642B	Byte in den PAB und Ende

Bei den beiden anderen Dateitypen schreibt der TI zu allererst die Bit für den Dateityp (FIXED oder VARIABLE, DISPLAY oder INTERNAL) in >834A.

Speicher enthält ein Datenfile:

G63DE ST	§>834A, §>834B	Enthält Bit FIX/VAR und DIS/INT
G63E1 AND	§>834A,>08	Bit DISPLAY/INTERNAL abspalten
G63E4 AND	§>834B,>10	Bit FIXED/VARIABLE abspalten
G63E7 SLL	§>834A,>01	An die richtige Stelle rücken
G63EA SRL	§>834B,>02	An die richtige Stelle rücken
G63ED OR	§>834A, §>834B	Beides in das EOF-Byte schreiben

Nun geht es für FIXED- und VARIABLE-Dateien getrennt weiter:

G63F0 CLOG	§>834B,>04	FIXED?
G63F3 BR	GROM§>641D	Nein: Sprung

Die in den folgenden beiden Routinen enthaltenen Abfragen haben bei beiden Dateitypen die gleiche Aufgabe. Der TI muß feststellen, ob das logische Dateiende erreicht ist, ob also die Datei noch weiter lesbar ist oder nicht. Außerdem, ob das physikalische Dateiende erreicht ist, ob also in die Datei noch geschrieben werden kann oder nicht. Für FIXED-Dateien sind diese Abfragen relativ umfangreich, da die REC-Nummer erst durch Multiplikation mit der

Satzlänge zu brauchbaren Pointern umgewandelt werden müssen.

Speicher enthält Datenfile im FIXED-Format:

G63F5	DINC	§>8350	REC-Nummer plus eins
G63F7	ST	§>834F,§>0003 (§>835A)	Logical Record Length (Satzlänge)
G63FD	CLR	§>834E	Hi-Byte löschen
G63FF	DST	§>834C,§>8350	REC-Nummer kopieren
G6402	DMUL	§>8350,§>834E	REC-Nummer mal Satzlänge nehmen
G6405	DCZ	§>8350	Überlauf beim Multiplizieren?
G6407	BR	GROM§>640E	Ja: Sprung
G6409	DCH	§>8352,§>835C	Ergebnis größer als Filelänge?
G640C	BR	GROM§>6413	Nein: weiter
G640E	OR	§>834A,>02	3. EOF-Bit: physikalisches Dateiende erreicht
G6411	BR	GROM§>641B	Byte schreiben und Ende
G6413	DCH	§>834C,§>0004 (§>835A)	REC-Nummer größer End-of-File-Pointer?
G6419	BS	GROM§>6428	Ja: logisches Dateiende erreicht
G641B	BR	GROM§>642B	Nein: Byte schreiben und Ende

Wesentlich einfacher sieht es da bei VARIABLE-Dateien aus. Allerdings fragt der TI nur ab, ob das logische Dateiende erreicht ist. Aus welchen Gründen Texas Instruments die Abfrage nach dem physikalischen Dateiende verschlampt hat, ist eventuell nur so zu erklären, daß es sehr unwahrscheinlich ist, daß die Datei einmal wirklich bis auf das letzte Byte vollgeschrieben wird. Meistens bleiben zum Schluß noch ein paar Byte übrig. Für die noch zu schreibenden Datensätze reicht das aber nicht, so daß bei jedem weiteren Schreibzugriff ein "Memory Full" erscheint, obwohl das physikalische Datei-Ende noch nicht voll erreicht ist und theoretisch noch Sätze geschrieben werden können.

Speicher enthält Datenfile im VARIABLE-Format:

G641D	DCHE	§>0006(§>835A), §>0004 (§>835A)	Current-File-Entry-Pointer größer/gleich End-of-File-Pointer?
G6426	BR	GROM§>642B	Nein: Byte schreiben und Ende
G6428	OR	§>834A,>01	2. EOF-Bit: logisches Dateiende

Nun schreibt der TI das Byte >834A in den PAB und kehrt zum Basic zurück.

EOF-Byte in den PAB schreiben:

G642B	ST	VDP§>FFFE (§>835A),§>834A	EOF-Byte in den PAB schreiben
G6431	BR	GROM§>61DE	GSR-Return to Basic

Damit ist diese Serie über die Programmierung einer DSR zu Ende. Ich hoffe, auch damit wieder einigen TI-Usern geholfen zu haben.

Sven Dyroff □

Wie wär's denn 'mal mit einem eigenen ROM?

Wie wäre es denn, wenn Sie die Möglichkeit hätten, sich ein eigenes ROM für den TI zu schreiben? Mit Ihrem eigenen Betriebssystem? Wenn Sie zu denjenigen gehören, die in Maschinensprache topfit sind, hardwaremäßig aber zu so einem Unterfangen nicht genug drauf haben, dann ist dieser Artikel goldrichtig für Sie.
Keine Angst, am Rechner selbst wird gar nicht viel gemacht. Ein Draht und eine Diode, das ist alles. Wenn Sie den TI aufschrauben, dann finden Sie in der rechten unteren Ecke der Platine ein waagerecht liegendes IC mit der Typenbezeichnung SN74LS32, das in den Schaltplänen des TI mit U507 bezeichnet ist. Zwischen diesem Chip und den darüber liegenden Scratch-Pad-RAMs ist ein kleiner Widerstand (R527) und 1KOhm (kann von Rechner zu Rechner variieren). An das linke Ende dieses Widerstandes löten Sie die Kathode einer einfachen Siliziumdiode an. Ihre Anode verbinden Sie mittels eines Drahtes mit Pin 38 des Ein-/Ausgabe-Bausteins TMS9901. Das ist der 40-Füßler, der sich links oben neben dem ebenfalls 40-füßigen Videoprozessors befindet. Anschließend schrauben Sie Ihren TI wieder zusammen.
Was Sie nun eigentlich gemacht haben, werden Sie fragen. Es gibt im TI den Hauptadreßdekoder (U504/SN74LS138), der den 64 KByte-Speicherbereich des TI in Blöcke zu je 8 KByte einteilt. Die unteren 8 KByte sind für das ROM, die folgenden 8 für die Low-Mem, und so weiter.
Die Freigabeleitung für die ROMs führt vom Hauptadreßdekoder über besagten Widerstand zum Chip-Select-Eingang (CS) der ROMs. Normalerweise führt die Leitung +5 V und die ROMs sind aus. Spricht der Prozessor sie an, dann geht der Pegel von +5 V auf 0V. Und eben das kann mit der Diode bei Bedarf verhindert werden. Der TMS 9901 hat nämlich noch drei CRU-Bit frei. Über das CRU-Bit 13 (Pin 29) können Daten eingelesen werden, wenn R319 (10 KOhm) entfernt wird, der den Pin des auf 1 (+5 V) legt. Und, was für Sie hier wichtig ist, über die CRU-Bit 16 und 17 (Pin 38 und 37) können Daten ausgegeben werden.

CRU-BIT 16 ALS SCHALTER

Wenn Sie jetzt also mit SBO 16 Pin 38 auf +5 V setzen, dann ist über die Diode die CS-Leitung der ROMs ständig auf +5 V, und es kann dort kein Zugriff mehr stattfinden. Der vorhandene Widerstand R527 verhindert dabei einen Kurzschluß. Da die Freigabeleitung aber nicht nur zu den ROMs geht, sondern auch zum Bus-Converter (der die 16 Bit der CPU auf 8 Bit runterholt), wird dieser nicht mehr bei ROM-

Zugriffen gesperrt.
Fazit: Sie können sich von einem Freund eine externe Schaltung bauen lassen, die exakt wie ein Peripheriegerät arbeitet. Nur, daß diese nun auf das CRU-Bit 16 reagieren und von ihm ein- und ausschaltbar sein muß und der Arbeitsbereich des EPROMs nicht bei >4000, sondern bei >0000 beginnt. Die Schaltung des Q-Save-Moduls aus Chip 8/85, Seite 48ff eignet sich sehr gut zum Umbau.
Für Profis, die sich die EPROMs in den Rechner direkt einlöten wollen: Beim Direkteinbau darf der Bus-Converter nicht, wie hier, im ROM-Bereich freigegeben werden. Er muß nach wie vor gesperrt sein, deshalb ist die oben beschriebene Methode nicht anwendbar. Statt dessen muß Pin 21 beider ROMs (CS2) von Masse gelöst und miteinander und mit Pin 38 des TMS9901 verbunden werden. Dazu muß ein Widerstand von 2.2 KOhm von dieser Verbindung nach Masse gelötet werden. Das wird am besten auf der Platinenunterseite gemacht, da Sie hier besser an die Kontakte kommen. Wenn dann noch CS2 der TI-ROMs auf diesem Pin aktiv low, die EPROMs aktiv high sind, ist die Schaltung fertig, und mit dem CRU-Bit 16 kann zwischen den TI-ROMs und den EPROMs hin- und hergeschaltet werden.
Sie sollten die externe Lösung nicht allzusehr belächeln, da sie einen nicht von der Hand zu weisenden Vorteil hat: Sie müssen den Rechner nicht nochmals aufschrauben, wenn Sie die EPROMs umbrennen wollen.
Hier noch ein kurzer Lageplan der Chips in der rechten unteren Platinenhälfte: *Sven Dyroff* □

```
 ---------------
!               !
! U610          !       ---
!               !      !   !
! ROM - HI      !      !   !
!               !      !   !
 ---------------       !U  !   ---     ---
                       !509!  !U  !   !U  !
                       !   !  !505!   !504!
                       !   !  !   !   !   !
 -----   -----   ----- !   !  !Neb!   !Hau!
!     ! !     ! !     !!   !  !ena!   !pta!
!     ! !     ! !     !  ---  !dre!   !dre!
!     ! !     ! !     !       !ßde!   !ßde!
!     ! !     ! !     !  ---  !kod!   !kod!
!U611 ! !U609 ! !U608 ! !   ! !er !   !er !
!     ! !     ! !     ! !   !  ---     ---
!     ! !     ! !     ! !   !
!     ! !Scrat! !Scrat! !U  !  ---     ---
! ROM ! !ch-Pa! !ch-Pa! !503! !   !   !   !
!     ! !d-Ram! !d-Ram! !   ! !   !   !   !
!     ! !     ! !     ! !   ! !   !   !   !
! LOW ! ! LOW ! ! HI  ! !   ! !U  !   !U  !
!     ! !     ! !     ! !   ! !506!   !508!
 -----   -----   -----   ---  !   !   !   !
                  ----        !   !   !   !
              --!RS27!--      !   !   !   !
                  ----        !   !   !   !
               ---------       ---     ---
              ! U507    !
              !SN74LS32 !
               ---------
```

Programm zum Ein- und Auslesen von Moduln im Bereich HEX 6000

Manche Moduln für den T99/4A verwenden keine GROM's, sondern nur den Speicherbereich bei >6000 bis >7FFF.
Mit diesem Programm kann man auf diesen Speicherbereich ohne Probleme zugreifen, sei es, um Software zu lesen oder abzuspeichern.
Der Vorteil dieses Speichers ist unter anderem der, daß man ihn mit einer Batteriepufferung ausstatten kann und somit wichtige Programme sofort zur Verfügung stehen, auch wenn der Computer zwischendurch ausgeschaltet wird.
Starten kann man das Programm mit dem Editor/Assembler. Möchte man nun ein File einlesen, benötigt man das GROM Simulationsmodul (siehe TI-REVUE Nr. 5/85), welches diesen Speicherbereich zur Verfügung stellt. In dem GROM-Bereich kann der Editor/Assembler liegen.
Zum Einlesen hat man entweder die Möglichkeit, nur eine Taste (Zahlen 1 bis 9) zu bedienen, dann wird ein Programm sofort geladen. Oder man gibt die Laufwerksnummer und den Programmnamen extra an.
Im Source Code wurden die Programme, die über die Tasten 1 bis 9 direkt geladen werden können,

```
**********************************
*                                *
* Lese- und Schreibroutine für    *
* Memory Image Programme in den Be-*
* reich Hex 6000.                *
*                                *
* Konfiguration: E/A, GROM Modul.  *
*                                *
**********************************
*
*
       DEF  LOAD
       REF  DSRLNK,VMBW,VMBR
       REF  KSCAN,VSBR,VSBW
*
PAB    DATA >0500,>1000,>0000     * Peripheral Access Block für die
       DATA >2000                 * DSR.
PABR   DATA >0600                 * PAB zum Abspeichern
PAB1X  DATA >0009                 * Namenslänge
       TEXT 'DSK1.PGM1'           * 4 Programme die von Laufwerk 1 geladen
PAB2X  DATA >0009                 * werden sollen.
       TEXT 'DSK1.PGM2'
PAB3X  DATA >0009
       TEXT 'DSK1.PGM3'
PAB4X  DATA >0009
       TEXT 'DSK1.PGM4'
*
TEX    TEXT '  Loader by Pieter Coates  1987 '  * Text für Bildschirmaufbau
TX     TEXT '                                '  * Leerzeile
       TEXT '    PROGRAM --> 6000            '
       TEXT '                                '
       TEXT '  DSK1.:     1  -> PGM1         '
       TEXT '             2  -> PGM2         '
       TEXT '             3  -> PGM3         '
       TEXT '             4  -> PGM4         '
       TEXT '             -  -> PGM5         '
       TEXT '             -  -> PGM6         '
       TEXT '             -  -> PGM7         '
       TEXT '             -  -> PGM8         '
       TEXT '                                '
       TEXT '             A  -> LOAD PRG     '
       TEXT '             S  -> SAVE PRG     '
       TEXT '             O  -> TITEL SCREEN '
*
ERR    TEXT '> ERROR <'                          * Text Meldungen
FINA   TEXT ' FILE NAME:'
REA    TEXT ' READY ? (Y/N)'
*
CURSOR  BYTE >F0,>90,>90,>90,>90,>90,>90,>F0   * Cursor Zeichen
*
        EVEN
*
BUFFER BSS >10                    * Buffer für Pab
MYWS   BSS >20                    * Eigener Workspace
*
LOAD   LWPI MYWS                  * Text Anzeigen
       LI R0,>00
       LI R1,TEX
       LI R2,>200
       BLWP $VMBW
*
       BL $LNCLR                  * Zeile für Meldungen löschen
*
       LI R0,>BF0                 * ASCII Wert 7E als Cursor definieren
       LI R1,CURSOR
       LI R2,>8
       BLWP $VMBW
*
       LI R5,>00FF                * Tastatur Abfrage
       MOV R5,$>8374              * Wert steht bei >8375
LOOP1  BLWP $KSCAN                * Kscan aufrufen
```

```
       MOV $>8374,R4
       CI R4,>0030                * 0 ?
       JEQ FINI
       CI R4,>0041                * A ?
       JEQ KEY
       CI R4,>0031                * 1 ?
       JEQ LOAD1
       CI R4,>0032                * 2 ?
       JEQ LOAD2
       CI R4,>0033                * 3 ?
       JEQ LOAD3
       CI R4,>0034                * 4 ?
       JEQ LOAD4
       CI R4,>0053                * S ?
       JEQ READJ
       JMP LOOP1
*
.OAD1 LI R1,PAB1X                 * Pointer von Programm 1 laden
      LI R2,>000B                 * Länge + 2
      JMP WORK
*
.OAD2 LI R1,PAB2X                 * Pointer von Programm 2 laden
      LI R2,>000B                 * Länge + 2
      JMP WORK
*
.OAD3 LI R1,PAB3X                 * Pointer von Programm 3 laden
      LI R2,>000B                 * Länge + 2
      JMP WORK
*
 OAD4 LI R1,PAB4X                 * Pointer von Programm 4 laden
      LI R2,>000B                 * Länge + 2
      JMP WORK
*
******************************
* Programm+Namen ins VDP und *
 DSR Routine ausführen       *
******************************

.ORK  LI R0,>0F08                 * Name ins VDP
      BLWP $VMBW
      LI  R0,>0F09
      MOV R0,$>8356               * PAB Byte 9 an >8356 übergeben

      BLWP $DSRLNK                * DSR aufrufen
      DATA >0008

      JEQ ERL                     * Error ?

      LI  R0,>1000                * Programm in >6000 laden
      LI  R1,>6000
      LI  R2,>2000
      BLWP $VMBR

      CLR $>837C                  *
INI   BLWP $>0000                 * QUIT !

EADJ BL $READ
      JMP KEY
      JMP LOAD

RL    BL $ERROR

OOP3 JMP LOAD

*******************
* Eingabe Routine *
*******************
*
KEY    LI R0,>0281                * 'FILE NAME' Anzeigen
       LI R1,FINA
       LI R2,>000B
       BLWP $VMBW
*
       LI R0,>028E                * Startaddresse, wo der Name
*                                   gelesen werden soll.
       LI R1,>7E00                * Cursor
MAX    DEC R0
       BLWP $VSBW
LOOP   CLR $>8374                 * Gesamte Tastatur soll geprüft werden
       BL $DELAY                  * Delay aufrufen
       DATA 8000                  * dez. 8000 Durchläufe
       BLWP $KSCAN
       MOV $>8374,R1
*
       CI R1,>000D                * Enter ?
       JEQ STOR
*
       CI R1,>0008                * FCTN S ?
       JEQ BACK
*
       CI R1,>00FF                * Wurde andere Taste bedient ?
       JEQ LOOP
*
MARK   SLA R1,>0008               * Byte verschieben
       BLWP $VSBW
       BL $DELAY                  * Verzögern
       DATA 4000
       INC R0
       CI R0,>029B                * Endaddresse schon erreicht ?
       JGT MAX1                   * Ja, -> MAX1
       LI R1,>7E00
       BLWP $VSBW
       JMP LOOP                   * Nein, nochmal
*
MAX1   DEC R0                     * Begrenzt Namenslänge
       JMP LOOP
*
BACK   BL $SPACE                  * Zeichen löschen 'FCTN S'
       CI R0,>028D
       JEQ LOOP
       DEC R0
       LI R1,>7E00
       BLWP $VSBW
       BL $SPACE
       JMP LOOP
*
STOR   BL $STOP
       JMP WORK
*
**********************************
* File Name lesen und in Buffer *
* schreiben.                     *
**********************************
*
STOP   CLR R1
       CLR R2
       LI R4,>2100
       LI R0,>028C                * Ab >028C wird gelesen
LESE   INC R0
       BLWP $VSBR
       INC R2
       CI R1,>7E00
       JEQ CURS
       CB R1,R4                   * Falls gelesenes Zeichen kleiner als >21
       JGT LESE                   * Lesevorgang beenden.
*
CURS   DEC R2
       MOV R2,$BUFFER             * Namenslänge in Buffer schreiben
*
       LI R0,>028D                * Name in Buffer schreiben
       LI R1,BUFFER
       INCT R1
       BLWP $VMBR
       LI R1,BUFFER               * R1 mit Buffer Pointer laden
       INCT R2
       RT
*
**********************************
* READ Routine zum Lesen ab  >6000 *
* und Abspeichern auf Diskette    *
**********************************
*
READ   LI R0,>0281                * READY (Y/N) auf >0281 ablegen
       LI R1,REA
       LI R2,>000E
       BLWP $VMBW
*
READ2 CLR R1                      * Tasten Abfrage
       BLWP $KSCAN
       MOV $>8374,R1
       CI R1,>0059                * Y ?
       JEQ READ1
       CI R1,>004E                * N ?
       JEQ LOOP3
       JMP READ2
*
READ1 LI R0,>0F00                 * PAB auf SAVE umschalten
       LI R1,PABR
       LI R2,>2
       BLWP $VMBW
*
       LI R0,>1000                * Programm von >6000 auf >V1000
       LI R1,>6000                * copieren. Länge 2000 Byte.
       LI R2,>2000
       BLWP $VMBW
*
       LI R0,>280                 * Zeile löschen
       LI R1,TX
       LI R2,>16
       BLWP $VMBW
       RT
*
LNCLR LI R0,>280                  * Unterprogramm; löscht Zeile >280
       LI R1,TX
       LI R2,>21
       BLWP $VMBW
       RT
*
SPACE MOV R0,R5                   * Unterprogramm; Zur Korrektur, setzt
       INC R0                     *               ' SPACE '.
       LI R1,>2000
       BLWP $VSBW
       MOV R5,R0
       RT
*
DELAY MOV *R11+,R15               * Unterprogramm; verzögert, um Tastenprellen
DEL    DEC R15                    * zu kompensieren. Variablenübergabe mit R11+.
       JOC DEL
       RT
*
ERROR LI R0,>223                  * Unterprogramm; gibt die Meldung ERROR aus.
       LI R1,ERR
       LI R2,>09
       BLWP $VMBW
       RT
*
       END LOAD
```

Die TI-REVUE sucht noch gute Autoren!

als PGM1, PGM2... et cetera, angegeben. Diese Namen können beliebig geändert werden.
Der Programmname darf jedoch nicht länger als 10 Buchstaben sein. Zusätzlich muß die Wortlänge als Hex-Zahl in der DATA-Direktive darüber angegeben werden (siehe Programmzeilen 8 bis 15)

Möchte man ein Modul auslesen, muß die Taste S für SAVE gedrückt werden.
Zunächst erscheint dann die Meldung READY (Y/N). Während diese Meldung erscheint, kann das auszulesende Modul in dem GROM-Port gesteckt werden.
Bedient man dann die Taste Y (Yes), wird man

gefragt, auf welches Laufwerk und mit welchem Namen das Programm abgespeichert werden soll.
Die Files sind alle im Memory Image-Format. Mit diesem einfachen Programm dürfte es keinem mehr schwerfallen, den >6000 Bereich ein- oder auszulesen.

Pieter Coates □

Programme unter X-Basic

Der im folgenden abgedruckte Loader für PROGRAM-Files (Assembler-Spiele und ähnliches) ermöglicht es im Gegensatz zu herkömmlichen Loadern, auch solche Programme unter XBasic zu fahren, deren Bildschirmausgabe speziell für das Fahren unter dem Editor/Assembler angepaßt wurde. Durch das Laden der VDP-Register und Verschieben des Zeichensatzes im VDP-RAM wird eine Umgebung geschaffen, die der des Editors/Assembler/Modules entspricht.
Es können lediglich jene Programme nicht zum Laufen gebracht werden, welche direkt auf Routinen zugreifen, die im Editor/Assembler programmiert sind (VSBW, VWTR, DSRLNK etc.).
Der Start des Programms erfolgt nach dem Laden automatisch nach etwa zwei Sekunden. Diese Zeit benötigt der Loader zum Initialisieren des VDP-RAMs, des RAM-PADs und der VDP-Register.
Aus dem beigefügten Assembler-Source-Listing ist auch zu ersehen, wie eine GPLLNK-Routine und eine DSRLNK-Routine für das XBasic-Modul programmiert werden können. Beide Routinen können auch dazu benutzt werden, völlig modulunabhängige Programme zu erstellen. Die GPLLNK-Routine läuft allerdings nur, wenn sie im Low-Memory (>2000–>3FFF) steht. Eine elegantere Lösung ist von Sven Dyroff in der CK Okt./Nov. 1987 veröffentlicht worden.

Mathias Eichhorn
TI-Workshop
Wiesbaden □

```
*         ************
*        **************
*        **          **
*        ** XLOADER2 **
*        **          **
*        **************
*         ************
*
*        (C) M. Eichhorn, Goethestr. 7, 6242 Kronberg/Ts.
*
*        Loader fuer PROGRAM-Files unter XBASIC
*        Benoetigte Konfiguration: XBASIC, 32K-RAM und Disk.
*
*        Aufruf des Programms mit:
*
*        CALL LINK("LOADER",Dateiname)
*
*
         AORG >24F4
*
         DEF  LOADER
*
STRREF EQU  >2014
VSBW   EQU  >2020
VMBW   EQU  >2024
VSBR   EQU  >2028
VMBR   EQU  >202C
VWTR   EQU  >2030
NAMLEN EQU  >8355
NAMPTR EQU  >8356
*
MYWS   BSS  32
*
NAMBUF DATA >0F00
       BSS  14
*
PABBUF DATA >0500          Opcode fuer LOAD
       DATA >1000          VDP-Buffer ab >1000
       DATA >0000
       DATA >2400          max. >2400 Bytes laden
       DATA >0000
*
B00    BYTE 0
B01    BYTE 1
*
       EVEN
*
LOADER LWPI MYWS           Eigenen Workspace laden
       CLR  R0
       CLR  R3
       LI   R1,1           Dateiname uebernehmen
       LI   R2,NAMBUF
       BLWP @STRREF
PABSET LI   R0,>0F00       PAB in VDP schreiben
       LI   R1,PABBUF
       LI   R2,>000A
       BLWP @VMBW
       LI   R0,>0F09
       LI   R1,NAMBUF
       LI   R2,>0010
       BLWP @VMBW
       LI   10,>0F09       Namenlaengenzeiger setzen
       MOV  10,@>8356
       BLWP @DSRLNK        File laden
       DATA 8
       LI   R0,>1000
       LI   R1,MYWS+12
       LI   R2,>0006
       BLWP @VMBR          Startparameter uebernehmen in R6, R7, R8
       LI   R0,>1006
       CI   R3,0
       JNE  LOAD1
       MOV  R8,R4          Startadresse sichern
LOAD1  INC  R3
       MOV  R8,R1          R8 --> Startadresse
       MOV  R7,R2          R7 --> Byteanzahl
       BLWP @VMBR          File von VDP in RAM schreiben
       MOV  R6,R6          R6 = 0?
       JEQ  START          Ja   --> DEF-Table setzen
       LI   R2,NAMBUF      Nein --> Letztes Byte des Filenamens +1
       MOVB *R2,R2
       SRL  R2,8
       AI   R2,NAMBUF
       AB   @B01,*R2
       JMP  PABSET         Naechstes File laden
*
VDPREG DATA >0000,>01E0,>0200,>030E
       DATA >0401,>0506,>0600,>07F5
*
START  LI   R1,VDPREG      VDP-Register setzen
       LI   R2,>0008
LOOP   MOV  *R1+,R0
       BLWP @VWTR
       DEC  R2
       JNE  LOOP
       LI   R0,>0300       VDP-RAM loeschen
       CLR  R1
       MOV  @>8370,R2      Hoechste Adresse im VDP-RAM
LOOP1  BLWP @VSBW
       INC  R0
       C    R0,R2
       JLE  LOOP1
       LI   R0,>0300     • Sprites ausschalten
       LI   R1,>D000
       BLWP @VSBW
*
       LI   R0,>3FFC       GPL-Ruecksprungdatas setzen (fuer GPLLNK)
       LI   R1,>0460       (Mnemonic fuer BRANCH)
       MOV  R1,*R0+
       LI   R1,GPLRT       (Ruecksprungadresse)
       MOV  R1,*R0
*
       LI   R0,>0900       Zeichensaetze laden
       MOV  R0,@>834A
       MOVB @B00,@>837C
       BLWP @GPLLNK
       DATA >0018
       LI   R0,>0B00
       MOV  R0,@>834A
       MOVB @B00,@>837C
       BLWP @GPLLNK
       DATA >004A
       LI   R0,>8300
       CLR  R1
       LI   R2,>0038
       BL   @CLRPAD
       LI   R0,>8380
       LI   R2,>0020
       BL   @CLRPAD
       B    *R4           Start des Programms
*
CLRPAD MOV  R1,*R0+       PAD loeschen
       DEC  R2
       JNE  CLRPAD
       RT
*
GPLLNK DATA SUBWS,GPL1
GRMADR BSS  2
*
GPL1   MOV  *R14+,@>83EC
       MOVB @>9802,@GRMADR
       MOVB @>9802,@GRMADR+1
       DEC  @GRMADR
       LWPI >83E0
       LI   R13,>9800
       MOVB @>8373,R4
       SRL  R4,8
       AI   R4,>8302
       LI   R5,>3FFA
       MOV  R5,*R4+
       LI   R5,>003A
```

```
       MOV  R5,*R4
       SWPB R4
       MOVB R4,@>8373
       B    @>0060
GPLRT  DECT @>8373
       LWPI SUBWS
       MOVB @GRMADR,@>9C02
       MOVB @GRMADR+1,@>9C02
       RTWP
*
DSRLNK DATA SUBWS,DSR1
*
SUBWS  BSS  32
*
DSRMSK DATA >2000
DSRBUF BSS  8
POINT  TEXT '.'
HAA    BYTE >AA
*
DSR1   MOV  *R14+,R5      Data fuer DSRLNK uebernehmen
       SZCB @DSRMSK,R15
       MOV  @>8356,R0     Zeiger auf Namenlaenge im PAB
       MOV  R0,R9
       AI   R9,>FFF8      Zeiger auf Beginn PAB
       BLWP @VSBR
       MOVB R1,R3
       SRL  R3,8          Namenlaenge in R3
       SETO R4
       LI   R2,DSRBUF
DSR2   INC  R0            Naechstes Byte des Namens
       INC  R4
       C    R4,R3         Ganzer Name verglichen?
       JEQ  DSR3          Ja --> weiter
       BLWP @VSBR
       MOVB R1,*R2+       1 Byte in aus PAB in Buffer
       CB   R1,@POINT     Byte = '.'?
       JNE  DSR2          Nein --> weiter
DSR3   MOV  R4,R4         Devicenamenlaenge = 0?
       JEQ  DSRER1
       CI   R4,7          Devicenamenlaenge groesser 7?
       JGT  DSRER1
       CLR  @>83D0        ROM-Searchpointer loeschen
       MOV  R4,@>8354     Pointer fuer DSR (Devicenamenlaenge)
       INC  R4
       A    R4,@>8356     Pointer fuer DSR (1. Byte nach Devicenamen)
       LWPI >83E0         GPL-Workspace laden
       CLR  R1            DSR-Nummer loeschen
       LI   R12,>0F00     Zeiger fuer Suche nach DSR
NOROM  MOV  R12,R12
       JEQ  ROMON
       SBZ  0             Device ausschalten
ROMON  AI   R12,>0100     naechstes Device
       CLR  @>83D0        ROM-Searchpointer loeschen
       CI   R12,>2000     letztes Device erreicht?
       JEQ  DSRERR
       MOV  R12,@>83D0    ROM-Searchpointer
       SBO  0             Device einschalten
       LI   R2,>4000
       CB   *R2,@HAA      Headerbyte >AA vorhanden?
       JNE  NOROM         Nein --> kein ROM da
       AI   R2,8          Zeiger auf DSRs
       JMP  DSR5
DSR4   MOV  @>83D2,R2     naechste DSR-Startadresse holen
       SBO  0             Device einschalten
DSR5   MOV  *R2,R2        Zeiger auf naechste DSR holen und ...
       JEQ  NOROM         Adresse 0 --> keine DSR
       MOV  R2,@>83D2     ... sichern
       INCT R2
       MOV  *R2+,R9       DSR-Startadresse
       MOVB @>8355,R5     Device/Namenlaenge holen
       JEQ  DSR7
       CB   R5,*R2+       Devicenamenlaenge vergleichen
       JNE  DSR4
       SRL  R5,8          Devicenamenlaenge (16 Bit)
       LI   R6,DSRBUF
DSR6   CB   *R6+,*R2+     Devicenamen vergleichen
       JNE  DSR4
       DEC  R5
       JNE  DSR6
DSR7   INC  R1            Nummer der DSR erhoehen
       BL   *R9           DSR anspringen
       JMP  DSR4          Ruecksp. ohne INCT R11: naechste DSR ausfuehren
       SBZ  0             normaler Ruecksprung: Device ausschalten
       LWPI SUBWS
       MOV  R9,R0
       BLWP @VSBR         PAB-Byte 1 lesen (Statusbyte)
       SRL  R1,13         Fehlercode in Bit 13-15
       JNE  DSRER2
       RTWP
DSRERR LWPI SUBWS         alten Workspace laden
DSRER1 CLR  R1            Fehler-Code 0 --> DSR nicht gefunden
DSRER2 SWPB R1
       MOVB R1,*R13       Fehlercode in R0
       SOCB @DSRMSK,R15   EQU-Bit im Statusregister (R15) setzen
       RTWP
*
       END
*
```

```
100 ! **********************
110 ! *                    *
120 ! * XLOADER  POKELISTE *
130 ! *                    *
140 ! *     M. EICHHORN    *
150 ! *  GOETHESTRASSE 7   *
160 ! * 6242 KRONBERG/TS.  *
170 ! *                    *
180 ! **********************
190 !
200 CALL INIT
210 ! Programm-Datas laden
220 FOR I=0 TO 85
230 READ A(1),A(2),A(3),A(4)
,A(5),A(6),A(7),A(8)
240 CALL LOAD(9460+I*8,A(1),
A(2),A(3),A(4),A(5),A(6),A(7
),A(8))
250 NEXT I
260 ! Def-Table setzen
270 CALL LOAD(16376,76,79,65
,68,69,82,37,48)
280 ! Pointer auf Def-Table
290 CALL LOAD(8196,63,248)
300 END
310 ! Datas
320 DATA 0,0,0,0,0,0,0,0,0,0
,0,0,203,20,203,53
330 DATA 203,78,203,231,204,
71,204,150,204,228,205,29,20
5,75,205,96
340 DATA 15,0,35,253,38,184,
40,183,41,182,42,195,43,193,
44,179
350 DATA 5,0,16,0,0,0,36,0,0
,0,0,1,2,224,36,244
360 DATA 4,192,4,195,2,1,0,1
,2,2,37,20,4,32,32,20
370 DATA 2,0,15,0,2,1,37,36,
2,2,0,10,4,32,32,36
380 DATA 2,0,15,9,2,1,37,20,
2,2,0,16,4,32,32,36
390 DATA 2,10,15,9,200,10,13
1,86,4,32,38,164,0,8,2,0
400 DATA 16,0,2,1,37,0,2,2,0
,6,4,32,32,44,2,0
410 DATA 16,6,2,131,0,0,22,1
,193,8,5,131,192,72,192,135
420 DATA 4,32,32,44,193,134,
19,17,2,2,37,20,208,146,9,13
0
430 DATA 2,34,37,20,180,160,
37,47,16,203,0,0,1,224,2,0
440 DATA 3,14,4,1,5,6,6,0,7,
245,2,1,37,174,2,2
450 DATA 0,8,192,49,4,32,32,
48,6,2,22,251,2,0,3,0
```

```
460 DATA 4,193,192,160,131,1
12,4,32,32,32,5,128,128,128,
18,251
470 DATA 2,0,3,0,2,1,208,0,4
,32,32,32,2,0,63,252
480 DATA 2,1,4,96,204,1,2,1,
38,142,196,1,2,0,9,0
490 DATA 200,0,131,74,216,32
,37,46,131,124,4,32,38,76,0,
24
500 DATA 2,0,11,0,200,0,131,
74,216,32,37,46,131,124,4,32
510 DATA 38,76,0,74,2,0,131,
0,4,193,2,2,0,56,6,160
520 DATA 38,68,2,0,131,128,2
,2,0,32,6,160,38,68,4,84
530 DATA 204,1,6,2,22,253,4,
91,38,168,38,82,17,5,200,62
540 DATA 131,236,216,32,152,
2,38,80,216,32,152,2,38,81,6
,32
550 DATA 38,80,2,224,131,224
,2,13,152,0,209,32,131,115,9
,132
560 DATA 2,36,131,2,2,5,63,2
50,205,5,2,5,0,58,197,5
570 DATA 6,196,216,4,131,115
,4,96,0,96,6,96,131,115,2,22
4
580 DATA 38,168,216,32,38,80
,156,2,216,32,38,81,156,2,3,
128
590 DATA 38,168,38,212,34,15
8,160,135,16,220,7,32,34,156
,6,160
600 DATA 36,58,6,1,128,193,1
7,6,2,34,255,170,4,224,34,15
2
610 DATA 4,96,35,16,32,0,22,
1,4,193,192,193,16,249,46,17
0
620 DATA 193,126,83,224,38,2
00,192,32,131,86,194,64,2,41
,255,248
630 DATA 4,32,32,40,208,193,
9,131,7,4,2,2,38,202,5,128
640 DATA 5,132,128,196,19,6,
4,32,32,40,220,129,152,1,38,
210
650 DATA 22,246,193,4,19,71,
2,132,0,7,21,68,4,224,131,20
8
660 DATA 200,4,131,84,5,132,
168,4,131,86,2,224,131,224,4
,193
670 DATA 2,12,15,0,195,12,19
,1,30,0,2,44,1,0,4,224
680 DATA 131,208,2,140,32,0,
19,44,200,12,131,208,29,0,2,
2
690 DATA 64,0,152,18,38,211,
22,238,2,34,0,8,16,3,192,160
700 DATA 131,210,29,0,192,14
6,19,230,200,2,131,210,5,194
,194,114
710 DATA 209,96,131,85,19,9,
156,133,22,242,9,133,2,6,38,
202
720 DATA 156,182,22,237,6,5,
22,252,5,129,6,153,16,232,30
,0
730 DATA 2,224,38,168,192,9,
4,32,32,40,9,209,22,4,3,128
740 DATA 2,224,38,168,4,193,
6,193,215,65,243,224,38,200,
3,128
```

Advanced Diagnostics - mehr als ein Disk-Editor

Mit dem 1985 erschienenen Programm Advanced Diagnostics, geschrieben von Steve Mildon, der bereits durch seinen Disk-Manager zum CorComp-9900 Diskcontroller bekannt wurde, steht dem TI eine Disketten-Utility zur Verfügung, nach der sich so mancher Besitzer anderer Rechner die Finger lecken würde. Ohne Übertreibung: DIAGS, wie es auch genannt wird, stellt eine neue Kategorie der Disketten-Editoren dar.

Advanced Diagnostics ist aber nicht nur ein Disketten Editor. Daneben kann man mit diesem Programm Disketten kopieren, Laufwerke testen, Disketteninhalte auf ihre logisch korrekte Struktur überprüfen und, ganz nebenbei, die wichtigsten Speicherbereiche des Rechners prüfen.

Die derzeit günstigste Bezugsquelle für DIAGS ist TENEX in USA, wo DIAGS für 29,95 US-Dollar angeboten wird (Katalog Sommer 1987), was beim derzeitigen Dollarkurs etwa DM 50,– sein dürften.

Zusammen mit der Diskette (Format einseitig, einfache Dichte), erhält der Käufer ein 34seitiges Handbuch in einem Format, das wohl das amerikanische Äquivalent zu DIN A5 sein soll, sowie einen großen Zettel, der auf letzte Änderungen im Programm nach Fertigstellung des Manuals hinweist.

Dieses Handbuch verdient besondere Beachtung. Weniger wegen des Umfangs, sondern hauptsächlich aufgrund der ausführlichen Beschreibung aller 24 Befehle, über die DIAGS verfügt und der knappen, aber durchaus korrekten Erklärung dessen, was vor Erscheinen dieses Programms nur echten Profis bekannt war: der grundlegenden Struktur der Sektoren unter dem TI-DOS, dem physikalischen Aufbau einer Datenspur der Diskette, der prinzipiellen Arbeitsweise eines Floppy-Disk-Controller-Formatter Chips (FDC) und der verschiedenen Aufzeichnungsformate „Einfache Dichte“ und „Doppelte Dichte“.

Die diversen Detail-Fehler, die dieses Kapitel enthält, können das gute Gesamtbild jedoch nicht trüben.
Die auf den letzten Seiten der Anleitung enthaltenen Informationen sind noch immer gut genug, abgeschrieben und als Eigenleistung verkauft zu werden. So zuletzt geschehen in der TI-REVUE 8/87 auf den Seiten 26ff – erkennbar eben an der Übernahme dieser Detail-Fehler durch den „Autor“. Genau genommen sind solche Plagiate aber eine indirekte Anerkennung dessen, was abgeschrieben wurde, die Übernahme der Fehler das Indiz dafür, daß es nur fürs Abschreiben gereicht hat.
Die Programmdiskette selbst ist gegen Kopieren geschützt, wobei jedoch zu bemerken ist, daß DIAGS das erste (und leider einzige) kopiergeschützte Programm von Millers Graphics ist, das auch auf einem Atronic CPS99 voll funktionsfähig ist.

Auf der Diskette befinden sich folgende Files:

BOXFORMAT
Ein Kommandofile, das es erlaubt, ganze Diskettenboxen neu zu formatieren (wie DM 1000)

DEMO, DEMO-1 und DEMO-2
3 Kommandofiles, welche eine recht interessante Demonstration der Möglichkeiten von DIAGS geben.

DIAG und LOAD
Die Ladeprogramme für Extended Basic.

DIAGS
Der Loader für Editor/Assembler und Minimemory.

DIAGCONFIG
Das Konfigurationsfile, welches unmittelbar nach Start des Hauptprogrammes abgearbeitet wird.

DISKHEADER
Ein Kommandofile, das die Option Sweep Disk des DM 1000 simuliert.

DSKCOPY1–2
Kommandofile zum Kopieren einer SS/SD Diskette von Laufwerk 1 nach 2.

INSTALLMGR
Mit diesem Kommandofile kann der CorComp-Diskmanager nachträglich auf der DIAGS-Diskette installiert werden, was sonst bekanntermaßen nur beim Formatieren möglich ist. Die Originaldiskette weist hierzu an den entsprechenden Stellen Leersektoren auf.

R&WTEST, READTEST, SEEKTEST
Diverse Kommandofiles, mit denen sich verschiedene Funktionen von Laufwerken testen lassen.

Außer einem Pseudo-Programmfile, das die Seriennummer enthält, sind keine Files auf der Diskette; das eigentliche Hauptprogramm befindet sich versteckt auf den Spuren 25ff.
DIAGS kann unter Extended Basic, Minimemory und Editor/Assembler gestartet werden. Das Programm lädt innerhalb weniger Sekunden (bei XBasic kommt der DIS/FIX-80 Zuschlag dazu), präsentiert kurz eine Copyright-Meldung mit Versionsnummer (bisher ist nur Version 1.0 verfügbar) und Name des Autors. Unmittelbar danach lädt es ein Konfigurationsfile, mit dem der Anwender in der Lage ist, die Besonderheiten seines eigenen Systems (Laufwerknummer, Drukker, Schrittzeiten der Laufwerke, Bildschirmfarben) gleich zu Anfang automatisch einzustellen. Dieses Konfigurationsfile,

DAS ARBEITEN MIT DIAGS

(Name DIAGCONFIG) kann mit einem Text-Editor erstellt (TI-Writer oder E/A-Editor) oder direkt im Sektor 177 (der Originaldiskette) manipuliert werden.

Wie oben bemerkt, verfügt DIAGS über 24 Befehle, die es versteht und ausführen kann. Dabei wird jedoch vom üblichen Verfahren der Zuordnung von Tasten und Befehlen abgewichen. DIAGS besitzt einen einfachen sogenannten Command Line Interpreter (CLI) oder auch Parser, der den in der Kommandozeile eingegebenen Text interpretiert. Die Befehle können im Klartext oder als Abkürzung eingegeben werden, gegebenenfalls gefolgt von den notwendigen Parametern. Die Liste aller Befehle läßt sich mit der Tastenkombination FCTN 7 als Fenster einblenden, wo auch eventuelle Parameter kurz beschrieben werden.
Der Benutzer kann einen oder mehrere Befehle in die Kommandozeile schreiben, die dann nacheinander ausgeführt werden. Dazu kommt die einzigartige Eingabemöglichkeit über ein sogenanntes Kommandofile. In diesem File, das, wie oben anhand der Konfigurationsdatei DIAGCONFIG beschrieben, mit einem Text-Editor erstellt werden kann, können sich genau die Eingaben befinden, die auch im Direktmodus zulässig sind. Zusammen mit einigen Steuersymbolen können so mehrfach zu wiederholende Eingaben jederzeit aus einem Kommandofile reproduziert werden.
Der Einfachheit halber gibt es aber trotzdem einige Funktionen, die mit einem einzelnen Tastendruck abgerufen werden können. Da ist zum Beispiel die bereits beschriebene Befehlseinblendung mit FCTN 7. Mit FCTN 0 (Null) kann jederzeit eine Hardcopy auf dem mit dem Befehl OUTPUT DEVICE (OD) spezifizierten Gerät erstellt werden. Das kann ein Drukker sein oder aber ein Diskettenfile, falls der Bildschirmausdruck später weiter verarbeitet werden soll.

Im Falle des Diskettenfiles kann vorher mit OUTPUT WIDTH (OW) festgelegt werden, ob es sich um ein DV80-File oder eines mit anderer Zeilenlänge handelt. Dabei ändert sich natürlich nichts daran, daß die Anzahl der Zeichen einer Zeile wie auf dem Bildschirm 32 beträgt.
FCTN 5 führt zu dem eingangs erwähnten Copyright-Titelbild und bei nochmaliger Betätigung zum TI-Titelbild.
DIAGS zeigt im Betrieb auf drei Fenster verschiedene Informationen an. Das obere, das „Status-Fenster“, zeigt ausschließlich laufwerkspezifische Daten wie Nummer des aktiven Laufwerks, momentan geschaltete Seite der Diskette, Nummer des Sektors und der Spur, in dem sich dieser Sektor befindet. Dazu kommt noch die Angabe von gewählter Step-Zeit, Anzahl freier, belegter und als fehlerhaft markierter Sektoren.
Das mittlere Fenster dient der Anzeige der Informationen, die DIAGS als Ergebnis eines bearbeiteten Befehls präsentiert. Die unteren beiden Zeilen schließlich sind dem CLI vorbehalten. Es können auf ein Mal so viele Befehle und Parameter eingegeben werden, wie in diese zwei Zeilen passen.

DER SEKTOREDITOR

Natürlich wird ein Programm wie DIAGS hauptsächlich als Disk-Editor eingesetzt werden. Der Sektoreditor von DIAGS weist jedoch einige Besonderheiten auf, die das Arbeiten mit ihm vereinfachen, aber auch eine Gefahr für die Diskette darstellen. Ein beliebiger Sektor wird mit dem Befehl EDIT SECTOR (ES), gefolgt von der Nummer des Sektors als Parameter, in den Kopierpuffer geladen, wo er editiert werden kann. Dabei unterscheidet sich DIAGS kaum von anderen Pro-

grammen, außer daß es die Tasten FCTN 1 und FCTN 2, DELETE und INSERT unterstützt. Soweit also nichts Besonderes, da auch eine Fortschaltung der Sektoren zu höheren oder niedrigeren Nummern mit FCTN 4 und FCTN 6 erfolgen kann. Leider ist es aber schnell passiert, daß der Anwender mit FCTN 9 (BACK) den Editor verläßt und den alten Befehl ES in WS ändert, um den Sektor zurückzuschreiben. Wurde aber die Sektorfortschaltung benutzt, dann wird der Sektor an die falsche Stelle geschrieben, da in der Kommandozeile noch die alte Sektornummer steht. Hier ist Vorsicht angebracht!

LISTE DER BEFEHLE

Da DIAGS jederzeit die komplette Befehlsliste einblenden kann, sollen hier nur die Befehle aufgeführt werden, die sich nicht allein durch ihren Namen erklären. Dabei wird in Klammern das Befehlskürzel angegeben.

CHECK DISK (CD):
DIAGS liest Sektor 0 und 1 der Diskette im gewählten Laufwerk und ermittelt die Anzahl der freien und belegten Sektoren. Dann prüft es anhand jedes einzelnen File-Descriptor-Sektors (das sind die Sektoren, deren Nummern in Sektor 1 stehen), ob jeder belegte Sektor von einem File regulär abgedeckt ist. Sektoren, die keinem File zugeordnet sind, werden als "Sectors mapped bad" angezeigt. Dazu erscheint anschließend eine Liste der Files, die nicht zusammenhängend vorliegen ("Fractured Files").

FIND FILE (FF):
Dieses Kommando benötigt als Parameter den Namen des Files, das gesucht werden soll. Dabei muß der Gerätename weggelassen werden (beispielsweise DSK1.). Als Ergebnis wird im oberen Fenster angezeigt, in welchem Sektor sich der File-Descriptor-Block des gesuchten Files befindet. Das mittlere Fenster informiert darüber, von welchem Typ das File ist und in welchen Abschnitten der Diskette seine Daten abgelegt sind.

HEAD STEP (HS):
Dient zur Einstellung der Wartezeit, die verstreichen muß, nachdem der Kopf des Diskettenlaufwerks um eine Spur nach innen oder außen gefahren wurde. Je geringer diese Zeit ist, um so schneller erfolgt der Zugriff auf Daten, die sich auf der Diskette an verschiedenen Orten befinden. Dieser Befehl erlaubt dann, wenn ein TI-Controller erkannt wurde, die Werte 6, 12, 20 und 30. Wurde während des Ladevorgangs ein anderer Controller erkannt, so nimmt DIAGS einen CorComp-9900 Controller an und erlaubt die Werte 3, 6, 10 und 15. Alle Angaben haben die Einheit Millisekunden.
Hier kommen wir gleich zum ersten Fehler von DIAGS. Die CorComp-Step-Zeiten gelten nur auf den Controllern, die mit dem CorComp-Etikett aus Anaheim versehen sind, also der Vorversion, die CorComp seinerzeit soviel Ärger machte. In diesen wird ein 2793-02 Chip eingesetzt, der diese kurzen Schrittzeiten möglich macht. Alle neueren CorComp-Controller (schwarzes oder gar kein Gehäuse) produzieren die gleichen Step-Zeiten wie der TI-Controller, auch wenn DIAGS bei diesen nur die Hälfte anzeigt. Die gleichen Step-Zeiten treten auch beim CPS99 und den Atronic-Controllern für die P-Box auf.

MOTOR SPEED (MS):
Hiermit kann (oder besser: könnte) man überprüfen, ob die Umdrehungsgeschwindigkeit der Laufwerkmotoren korrekt ist. Als Zeitbasis wird der Interrupt des Video-Prozessors benutzt, dessen Frequenz auf den europäischen Konsolen (PAL-Norm) 50 Hertz beträgt. Dieser Wert wird als zweiter Parameter benötigt. DIAGS ermittelt nun die Anzahl der VDP-Interrupts, die zwischen dem zweimaligen Auftauchen des Index-Pulses vom Laufwerk auftreten. Aus dieser Interruptanzahl kann die Drehzahl des Laufwerkmotors berechnet werden. Es ist aber besser, wenn die Zeit für mehrere Diskettenumdrehungen gemessen wird. Im ersten Parameter wird daher angegeben, wieviele Umdrehungen ausgezählt werden sollen, bevor ein Meßwert ausgegeben wird.
Die Anzeige der Drehzahl erfolgt auf einem Farbbalken, in dem ein dreieckiges Sprite wie ein Zeiger hin und her läuft. Was hier angezeigt wird, ist aber ziemlich nutzlos, da bei DIAGS eigentlich alle Laufwerke zu langsam laufen. Eine zu geringe Drehzahl ist selten problematisch, eine zu hohe jedoch immer. Zeigt DIAGS den Nominalwert von 300 Umdrehungen an, so ist es wahrscheinlich, daß dieses Laufwerk nicht in doppelter Dichte formatieren kann, da die Spurlängen-Toleranz dann geringer ist. Der MS-Option sollte daher kein allzu großes Vertrauen geschenkt werden.

READ TRACK (RT):
Hier trennt sich die Spreu vom Weizen. Sollte eine Version des Advanced Diagnostics bei diesem Befehl streiken, so benutzt man eine Raubkopie. Entgegen anderslautenden Angaben sind alle Spurbefehle, also READ und WRITE sowie SEEK TRACK, auch auf Atronic-Controllern und CPS-99 funktionsfähig. Leider wurde der READ TRACK Befehl etwas halbherzig realisiert, um dem Programm nicht direkt zu erlauben, in die geschützten Sektoren zu sehen. Daher kommt es zum Beispiel, daß das SEEKTEST-Kommandofile nicht mit der Originaldiskette arbeitet, weil manche Spuren falsche Kennungen besitzen. Leere oder defekte Spuren können deshalb nur in Ausnahmefällen gelesen werden.
READ TRACK liest im Normalfall eine ganze Spur in den Kopierpuffer, in dem man sie sich dann ansehen kann. Hier läßt sich ersehen, welche Daten zwischen den Sektoren dafür sorgen, daß jeder einzelne Sektor korrekt identifiziert werden kann. Leider sind die Möglichkeiten dieses Befehls beschränkt, da nur ganz normale Spuren gelesen, jedoch nicht editiert und zurückgeschrieben werden können. Die Bezeichnung Spureditor ist daher etwas schmeichelhaft.

WRITE TRACK (WT):
Der vielleicht leistungsfähigste Befehl von DIAGS. Mit ihm ist es möglich, eine Diskette spurweise zu formatieren, wobei man wählen kann, welche Spur und welche Seite in welcher Dichte formatiert werden soll. Die interessanteste Anwendung kann dieser Befehl finden, wenn ehemals einseitige Disketten ohne Datenverlust auf zweiseitige „umgestrickt" werden sollen. Hier kommt dann auch ein Kommandofile zum besten Einsatz. Da DIAGS alle Spurnummern über 39 als Spuren auf der Seite 2 ansieht, kann durch die Befehlsfolge WT 40 S WT 41 S WT 42 S... WT 79 S eine Diskette im Format Einseitig/Einfache Dichte auf Zweiseitig/Einfache Dichte gebracht werden (das ‚S' steht für Single Density). Anschließend wird noch Sektor 0 angepaßt (von Hand), und die Operation ist beendet.
Leider ist auch der WT-Befehl in seiner Leistungsfähigkeit beschränkt, da er nie den Spurpufferinhalt schreibt, sondern immer eine Spur mit norma-

len Formatierungsdaten herstellt. Zwar kann wahlweise jede Spur in einer anderen Speicherdichte formatiert werden, doch bringt man sich damit nur das DOS des Controllers durcheinander, wenn man eine solche Diskette benutzt.

USE DSR (UD):
Dieser Befehl erlaubt es DIAGS, bei Sektoroperationen die DSR-Software von Diskcontrollern oder RAM-Disks zu benutzen, statt, wie sonst, eigene Software. Damit kann der Zwang aufgehoben werden, daß Spuren über 39 auf der Seite 2 der Diskette angenommen werden. So können dann auch 80-Spur-Laufwerke angesprochen werden, wenn entsprechende DSR-Software im Controller steckt. Auch RAM-Disks können auf Sektorebene angesprochen werden, wenn sie die entsprechenden Routinen (Name >10, Sektor-I/O) unterstützen. Leider gibt es nur diesen einen Befehl, um das DSR-ROM zu aktivieren oder zu deaktivieren. Besonders nachteilig wirkt sich dies in einem Kommandofile aus, da es nicht eindeutig möglich ist, das UD-Flag zu setzen oder zu löschen.

DAS KOMMANDOFILE

Wie bereits erwähnt, kann mit einem Text-Editor ein Kommandofile erstellt werden, das die Eingaben in der Kommandozeile simuliert. Komfortabler als mit E/A geht das natürlich mit TI-Writer, doch muß hier darauf geachtet werden, daß das File mit der Option DD (Datei Drucken) abgespeichert wird, um DIAGS mit den Tabulatorcodes nicht durcheinander zu bringen.
Die Länge eines Kommandofiles ist auf 2 KByte begrenzt, was bei etwa neun Sektoren Filelänge erreicht wird. Längere Files werden von DIAGS ignoriert. Dabei sollte beachtet werden, daß eine Zeile Text nur so lang sein darf, daß sie komplett in die Kommandozeile paßt, also nicht mehr als 52 Zeichen enthält.
Die Platzbegrenzung auf 2 KByte ist aber nicht weiter schlimm, da sich Kommandofiles untereinander aufrufen können.
Neben allen Kommandos in direkter Eingabe sind folgende Sonderzeichen erlaubt, die alle einzeln in eckige Klammern gesetzt werden müssen (im deutschen Zeichensatz Ä und Ü, weshalb sie hier weggelassen werden) und die es erlauben, alle Einzeltastenbefehle des Direktmodus' auszulösen.
3, 4 und 7 simulieren die Tasten FCTN 1 (Zeichen löschen), FCTN 2 (Zeichen einfügen) und FCTN 3 (Zeile oder Sektorinhalt löschen) im Direktmodus.
2, 14 und 12 simulieren FCTN 4 (nächsten Sektor lesen), FCTN 5 (Sprung zum Programmtitelbild) und FCTN 6 (vorhergehenden Sektor lesen).
1, 6 und 15 bewirken die Einblendung des Hilfsmenüs (FCTN 7), die erneute Prüfung einer als defekt gemeldeten Speicherstelle (FCTN 8) oder das Verlassen des Sektor- oder Spureditors.
188 fertigt auch im Kommandofile eine Bildschirmhardcopy an (FCTN 0),
5 erlaubt die Umschaltung zwischen ASCII- und Hex-Darstellung (FCTN =).
11, 10, 8 sowie 9 simulieren die Cursortasten FCTN E, X, S und D. Die Controlcodes E und X für das Blättern im Spureditor und im Hilfsmenü werden im Kommandofile mit 133 und 152 gesteuert.
Dazu kommen folgende Sonderzeichen, die nur in einem Kommandofile akzeptiert werden:
253 bewirkt eine Pause von etwa einer Sekunde, 254 die langsame Ausführung des Kommandofiles (zum Zuschauen), 255 sorgt für schnelles Arbeiten und der Leerstring schließlich, also eckige Klammer auf, direkt gefolgt von der eckigen Klammer zu, zeigt das Ende eines Kommandofiles an.

BESONDERHEITEN

DIAGS erlaubt den Zugriff bis auf die unterste Ebene der Diskettenstruktur: das Lesen ganzer Spurinhalte. Dies ist mit Den im Diskcontroller eingebauten Routinen nicht realisierbar, weshalb dafür eigene Software vorhanden ist. Hierzu muß DIAGS jedoch die besondere Hardware des TI-Controllers beachten. Daher wird beim Start des Programms zuerst der Controllertyp ermittelt, da auch die geschützten Sektoren mit eigenen Programm geladen werden müssen.
Bei dieser Controller-Erkennung geht DIAGS recht trickreich vor, indem der Befehl CALL MGR im DSR-ROM des Controllers gesucht wird. Wurde dieser gefunden, dann handelt es sich nicht um einen TI-Controller. Da auch die Atronic-Disk-Systeme diesen Befehl erlauben und zudem in der Hardware dem CorComp-Controller sehr ähnlich sind, kann die Originalversion von DIAGS auch auf diesen deutschen Controllern benutzt werden. Es gibt einen dritten, in der Hardware völlig verschiedenen, Diskcontroller von Myarc. Mit diesem System arbeitet DIAGS nicht.

RESÜMEE

Durch die Vielzahl der verfügbaren Funktionen mag es zuerst so aussehen, daß DIAGS schwer zu bedienen sei. Dazu mag noch das englischsprachige Manual seinen Teil beitragen. Doch die englischen Kommandos wird man sich ebenso wie die des Basic aneignen können. Was die Anleitung angeht, so sollte dieser Artikel eine Brücke zum umfangreichen Manual dieses außergewöhnlichen Programms bauen.

Christopher Winter, system-99 User-Group □

Dialog: Turbopasc 99 und Basic-Programme editieren

Zum Erfahrungsbericht zu TurboPasc '99 in der TI-REVUE 10/87 erhielten wir einen Leserbrief von Diethelm Schädler, den wir Ihnen auf gar keinen Fall vorenthalten wollen. Da auch wir nicht alles wissen können, freuen wir uns, wenn unsere Leser uns und damit auch andere TI-User unterstützen. Wir wollen so versuchen, den TI'lern weiterzuhelfen, die ja von Texas Instruments und jetzt anscheinend auch von den verbliebenen Firmen, die noch mit dem TI99 Geld verdienen wollen, nicht gerade mit Infos verwöhnt worden sind. Wir würden uns freuen, wenn wir in der Zukunft verstärkt ein Bindeglied zwischen den TI-Usern sein könnten. Wenn Sie Fragen zu Ihrem TI99 haben, oder aber, wenn Sie Lösungen

und Antworten hierzu anbieten können, so schreiben Sie uns. Wir wollen uns bemühen, alle Fragen soweit als möglich, entweder hier in der TI-REVUE oder aber auch direkt per Post, zu beantworten.

Herr Schädler schreibt:

Ich möchte die Frage von Uwe Schulze (TurboPasc '99 und RELATIVE Dateien) in der HCA Nr. 12/1 beantworten. Es kommt mir allerdings recht eigenartig vor, daß Softpoint nicht antworten konnte/wollte, da ich vor ein paar Monaten diese Firma schriftlich auf diesen Programmierfehler hingewiesen und auch eine Möglichkeit, dieses Problem zu umgehen, geschildert habe! Darauf habe ich noch nicht einmal eine Antwort erhalten.

Zur Antwort:

Zuerst müssen Sie überlegen, wieviel Datensätze Sie benötigen. Erhöhen Sie diese Zahl um eins. Nach jedem Schreibvorgang beschreiben Sie auch noch einmal den Record mit der oben berechneten Recordnummer (maximale Anzahl Datensätze +1).

Wenn Sie Datensätze lesen, müssen Sie nur noch darauf achten, daß diese auch existieren.

Am besten initialisieren Sie alle Records mit einem Wert, der einen „leeren" Record anzeigt. Die Datei sollten Sie entweder vor jedem Schreiben/Lesen öffnen und gleich anschließend wieder schließen, oder Sie öffnen die Datei gleich zweimal (einmal als Lese-Datei und einmal als Schreib-Datei) mit unterschiedlichen Namen für die File-Variablen. Letztere Möglichkeit habe ich aber noch nicht ausprobiert.

Grund:

Der Recordzeiger wird auf der Diskette auf die zuletzt geschriebene Recordnummer gesetzt. Wenn Sie also einen existierenden Record mit höherer Nummer lesen wollen, so ist er scheinbar nicht existent.

Da ich glaube, noch etliche Fragen zu Turbo-Pasc '99 beantworten zu können, hier meine Adresse:

Diethelm Schädler,
Kranichstraße 11,
6700 Ludwigshafen/Rh.

Es sollte aber ein adressierter und frankierter Rückumschlag beiliegen.

Auch zum Leserbrief von Dr. Volker Fuchs (Basic-Programme editieren) in der TI-REVUE 10/87 erhielten wir eine Antwort. Hierin wird eine, allerdings etwas umständliche, Möglichkeit aufgezeigt, ein mit dem Editor/Assembler oder TI-Writer geschriebenes Basic-Programm aus dem DIS/VAR80-Format ins PROGRAM-Format zu konvertieren. Für diejenigen, die nicht das CONVERT-Programm besitzen, dürfte dies eine Hilfe sein.

KONVERT-PROGRAMM EXISTIERT

Im Dialogteil der TI-REVUE 10/87 wurde unter anderem die Frage von Herrn Dr. Fuchs aus Freiburg beantwortet. Dabei fiel mir ein, daß in meinem Programmarchiv ein CONVERT-Programm existiert. Benötigt werden das Extended-Basic-Modul und die für den TI-Writer oder Editor/Assembler notwendigen Erweiterungen. Das Programm erstellt ein MERGE-File im DIS/VAR163-Format, welches anschließend mit dem XB-Modul weiterverarbeitet werden kann. Die Handhabung ist sehr einfach. Der Autor ist mir unbekannt. Bei Zusendung eines frankierten Rückumschlages sowie einer Leerdiskette bin ich bereit, das Programm weiterzugeben.

Meine Adresse:

Norbert Bleuel,
Rüdesheimer Straße 110,
6100 Darmstadt

TI99/4A

ACHTUNG, TI99/4A-Freunde! Verkaufe: 32KB-Speichererw. m. 16-Bit-Datenbus, seitl. ansteckbar 90,– DM. Module: Minus-Mission 5,– DM; Soccer 20,– DM; Othello 20,– DM. Tel. 04187/7303, Axel Papf

SENSATIONELLE PREISE wg. Systemwechsel! Viele nützliche Programme u. gute Spiele – größtenteils orig. TI – auf Disk., Kass. u. Modulen sowie Fachbücher äußerst preisgünstig zu verkaufen. Alles neuwertig! Liste anfordern bei: Rudhard Thielmann, Groschenweg 29, 5000 Köln 40, Tel. 02234/79646 abends

Suche „Mouse" für TI sowie funktionierenden Lightpen, od. jemanden, der mir einen baut. Beides im Tausch gg. Assembler-Programme aller Art, Anzahl egal. Frank Müller, Ursfelderstr. 49, 5014 Kerpen-Türnich, Tel. 02237/8278

Verkaufe TI-Laufwerk, intern; VB: DM 150,–. Ebenso XB II+, VB DM 130,:. Tel. 07191/58233 (abends)

VERKAUFE, alles orig. TI: RS 232 intern DM 170,–; 32KB-RAM intern DM 220,–; TERM. EMULATOR II DM 50,–; BIT-MAC Zeichenprog. DM 60,–; Bücher: XBasic f. Fortgeschrittene DM 30,–; Assembl.-Kurs ASEM 1 DM 40,–. W. Zacher, Tel. 089/849244

VERKAUFE KOMPLETTANLAGE: TI99/4A, P-Box, 2 Laufwerke, Sprachsynth., Lit., Supermodul, Module auf Disk., viel Software f. DM 2.200,– VB. Joysticks usw. G. Schilder, Eilperstr. 30, 5800 Hagen 1, Tel. ab 17 Uhr 02331/71153

Verkaufe TI99/4A, Module: Parsec, Indoor Soccer, Datenverarb. Ohne Joystick, m. Datenrec. u. Anschlußkabel. VB DM 150,– per Nachnahme. Anfragen bei: Thomas Weber, Leipziger Ring 13, 6638 Dillingen, Tel. 06831/71864

Suche Data Becker „Computer-Schach-Buch" bzw. das Listing des Basic-Schach-Programms. Tausche gg. „jede" gewünschte SOFTWARE (Ass.). Auch andere Schachprogramm-Listings gesucht. Frank Müller, Ursfelderstr. 49, 5014 Kerpen-Türnich, Tel. 02237/8278

FAST GESCHENKT: TI99/4A-Reste wie Platinen, Tastaturen, HF-Mod., Trafos usw. f. 50,– DM; TI REVUE ab 1/84 30,– DM; Orig.-Schaltpläne u. IC-Datenblätter 20,– DM; über 50 Zeitschr. m. TI-Prog. ab 1983, auch amerik., f. 50,– DM. Tel. 02303/60360

Externe RS232 Schnittstelle zu verkaufen: 90,– DM. Tel. 07531/76178

TI99/4A m. RS232-Schnittstelle v. Atronic, Ex-Basic, Schach- u. Minus-Misson-Modul, TI- u. Ex-Basic-Lernkass., Othello-Mod., „TI-99 Tips u. Tricks" v. Data Becker, TI-Joysticks, Programme auf Kass. Kompl. f. DM 580,–. Robert Beck, Primelweg 5, 7770 Überlingen, Tel. 07551/66752

Suche RGB-Modulator f. TI99. Angebote bitte an Jürgen Münzer. Tel. 0721/385554

VERKAUFE: „Broker, das Börsenspiel" (benötigt den ganzen Speicher!) auf Disk. m. Anl. DM 15,–; TI-Artist V. 2.0, orig. USA, DM 30,–; The Killing (super Rollenspiel um Magie) m. Anl. DM 10,–. Alles orig.-Prog. Sascha Will, Tel. 07972/431

VERKAUFE: Peri-Box DM 150,–; RS232-Karte DM 100,–; 32K-Erw.-Karte DM 100,–; Ex-Basic orig. TI DM 80,–; Bücher: 16-Bit-Mikroproz.-Kursbuch DM 10,–; 9900 Family Systems Design DM 15,–; alles zus. DM 350,–. Tel. 0761/494350

* TI99/4A * KOMPLETTANLAGE ZU VERKAUFEN, Expansion m. 512K-RAM-Disk, 192K-RAM-Disk, UCSD, Triple Tech, 2 RS232, TI-Contr., 32KB-RAM, Konsole, Monitor, Modul-Exp., jede Menge Softw. u. Bücher. NP DM 12.000,–, Verkauf VB. Tel. 089/849244 abends

Verkaufe Konsole, X-Basic, Box, RAM-Erw., Laufwerk DS, P-Code-Sys u. EM Assembler. Auch einzeln (soweit sinnvoll). R. Dillmann, 7602 Oberkirch, Tel. 07802/1464

*** TI99/4A *** VERKAUFE: MBI-Interface (Centr.), Module z.B. Schach, Bücher, Kass.-Softw. Listen gg. Porto bei Brose, Birkenstr. 19, 7259 Friolzheim

VERKAUFE wg. Systemaufgabe sämtl. Zubehör f. TI99/4A. Info bei Wolfgang Kätzlmeier, Erhartstr. 6, 8070 Ingolstadt

Suche CORCOM-Controller, Triple Teach-Card, Flexkabel, Interface-Karte. Bitte Tel. 0711/843775 anrufen

Der TI-Club Göttingen bietet TI-Usern in Südniedersachsen Freeware u. Kontakte. Info: 0551/71508

SUPER SONDERANGEBOT! Orig. TI RS232 (extern), Textdrucker (Teleprint), 32K-Speichererw. extern (Eigenbau) f. insgesamt 170,– DM zu verkaufen. Tel. 07148/7348

VERKAUFE: TI99/4A, XBasic, 2 Joysticks, Kass.-Rec., Parsec, TI Invaders, Startreck, Munchman, dt. Handbuch, ca. 50 Spiele auf Kass., TI REVUE v. 1/84 – 12/87, 4 TI REVUE Special, dopp. Kass.-Rec.-Kabel, zus. f. 350,– DM. Tel. 040/8323696 ab 18 Uhr

DEM TI UNTERS HEMD GESCHAUT (III)

Gewisse "Lötkolbenerfahrungen" müssen Sie schon mitbringen, jetzt geht es nämlich ans Eingemachte.

Nachdem wir die Hardware des TI 99/4A nun aus dem Eff-Eff kennen, sind wird in der Lage, sie eigenmächtig zu verändern. Das Hauptziel wird für jeden TI-USER zunächst die Erweiterung des vorhandenen CPU-Speicherraumes sein. Damit kann er größere Programme in BASIC schreiben: vor allem aber wird die Möglichkeit geschaffen, Maschinenprogramme zu schreiben und ablaufen zu lassen. Außerdem werden weitere Hardware-Veränderungen vorgenommen, deren Nutzen sich im Abschnitt 3 „Revolution" offenbaren wird.

2. EVOLUTION DER TI 99/4A - KONSOLE

Direkt in die Konsole wird eine Leiterplatte eingebaut, die die Funktionen

* 32-KByte-Speichererweiterung
* vollständige CRU-Dekodierung
* Ergänzungen zur Steuerelektronik

vornimmt. Die Schaltung der Leiterkarte ist in Abbildung 3, das Layout in Abbildung 4 dargestellt. Der Schwierigkeitsgrad ist, obwohl zweiseitig, nicht sehr hoch, so daß sie als nicht durchkontaktierte Leiterkarte hergestellt werden kann. Sie kann auch ohne Verwendung von Planfilm und fotochemischer Verfahren mit der (ruhigen) Hand gezeichnet werden.

Von der Zusatz-Leiterkarte werden zusätzliche Signale für die Erweiterungsbox erzeugt beziehungsweise von dieser empfangen. Daher wurde auch vom Original-TI-Steckverbinder-System abgegangen. Im Muster wurde ein 58poliger indirekter Steckverbinder zur Kopplung an das Verlängerungskabel verwendet, es kann aber auch mit anderen Steckern gearbeitet werden. Der indirekte Steckverbinder wurde auf die Steckkontakte des direkten Verbinders auf dem Konsolen-Print gelötet. Die Aussparung am TI-Gehäuse muß dazu etwas erweitert werden (Bild 1).

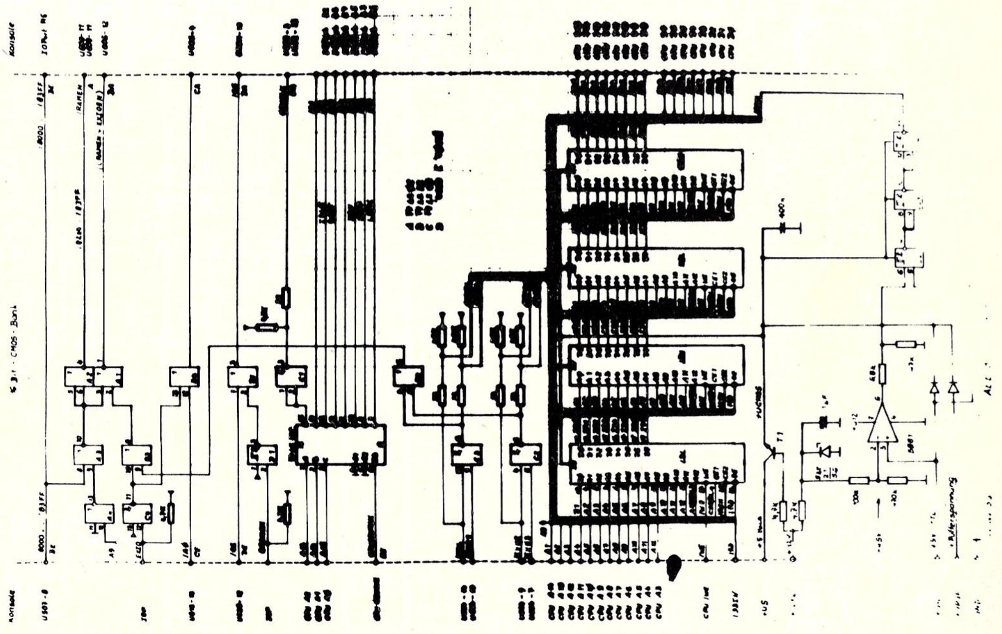
Für den Betrieb der Leiterkarte sind Änderungen im Konzept der TI 99/4A-Konsole notwendig. An sechs Punkten der im Abschnitt „Anatomie" beschriebenen Innenschaltung der Konsole sind Signalwege zu unterbrechen. Die Punkte werden mit Großbuchstaben A, B, C, D, E und F bezeichnet und sind in der Abbildung 1 aus dem Beitrag „Anatomie" eingezeichnet.

Um die Schwierigkeiten bei der CRU wegen der unvollständigen CRU-Adreß-Decodierung auszuräumen, wird die Leitung CRUCLK der CPU vor ihrer Verwendung aufgetrennt (Unterbrechung E in Abb. 1) und zur Zusatz-Leiterplatte zum Eingang EE geführt. CRUCLK wird auf den Eingang E3 (Pin 6) eines „1 aus 8-Decoders" 74LS138 gegeben. Die Eingänge /E1 und /E2 werden nach Masse gelegt und die Eingänge A bis C werden mit den drei höherwertigen Adreßbit A13 bis A15 belegt.

Am Ausgang Q0 steht dann der negierte CRUCLK zur Verfügung A 1/3. . A15 = 0), der nochmals invertiert werden muß (Gatter C1) und von der

2.1 AUFHEBEN DER CRU - SCHWIERIGKEITEN

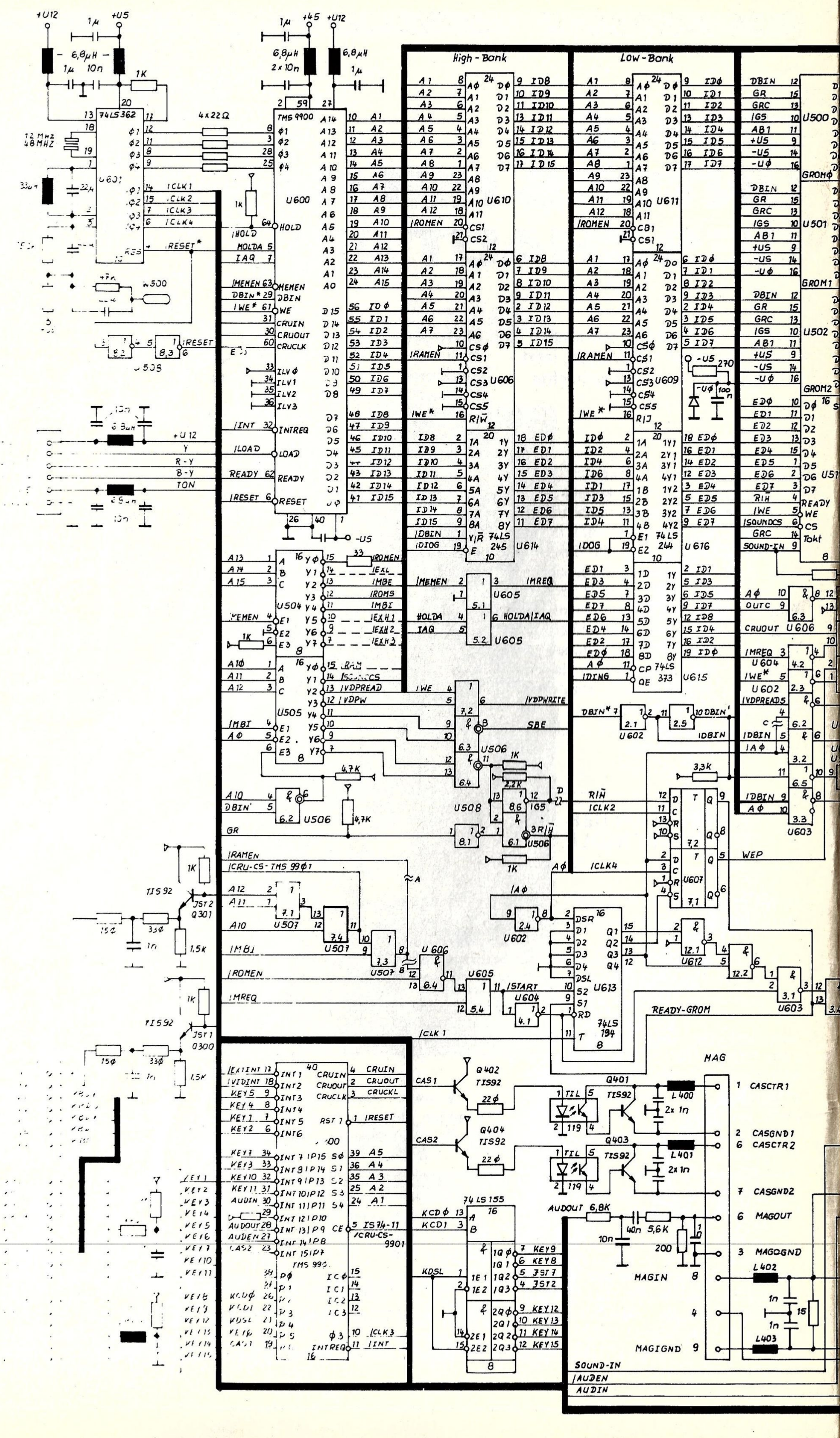
Zusatz-Leiterkarte über den Ausgang EA in die aufgetrennte Leitung von CRUCLK eingespeist wird. Die nachgeschaltete R-Kombination dient



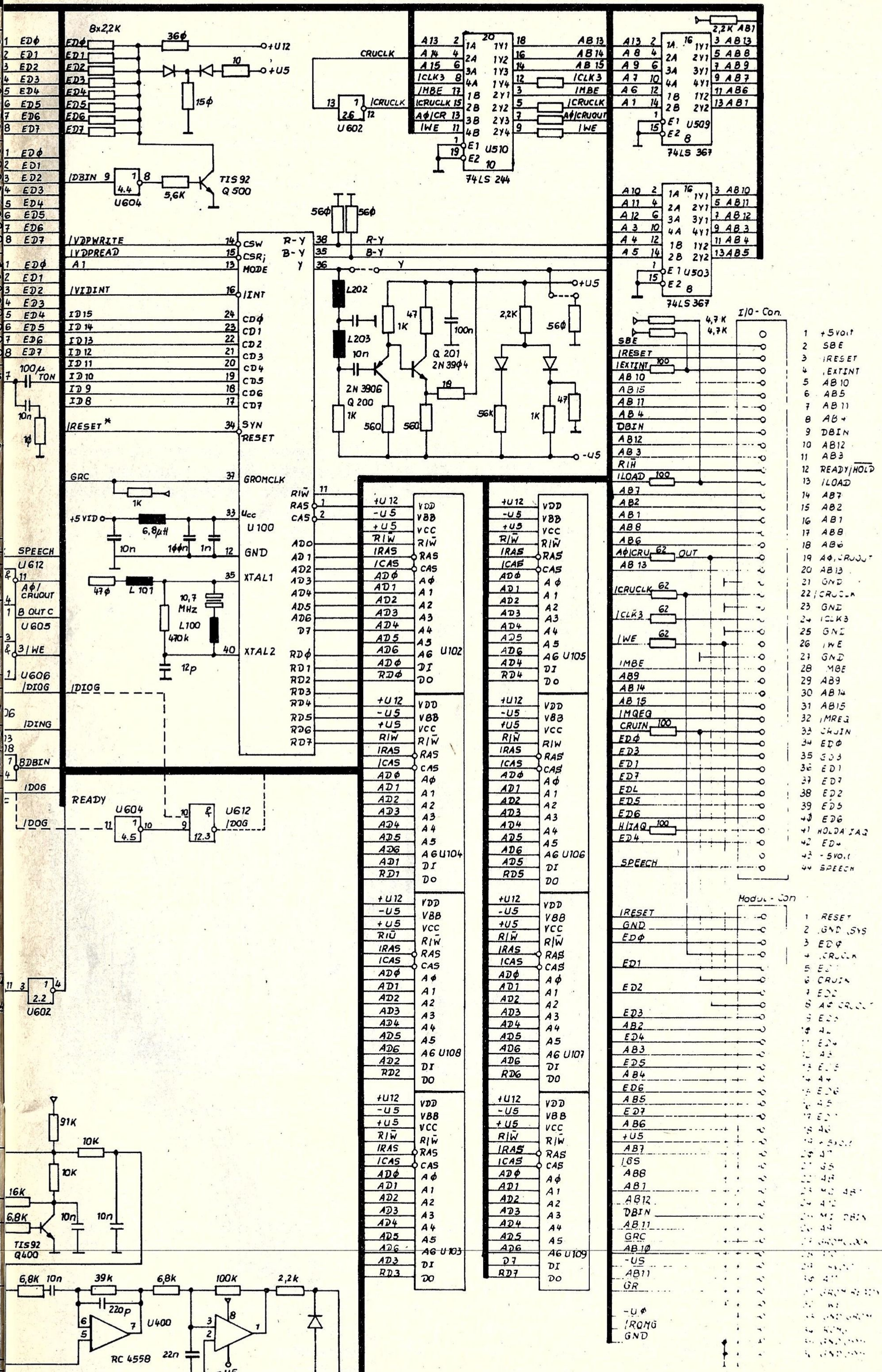
Der vergrösserte Schaltplan auf der nächsten Doppelseite (Im Text als Bild 1 bezeichnet) zeigt Ihnen, wo Sie ansetzen müssen

einer optimalen Anpassung an die Leitung.

Damit ist die CRU-Adreß-Decodierung „sauber" und es können alle Befehle des TMS 9900-Assemblers verwendet werden. Überdies stehen die von dem 74LS138 erzeugten fünf weiteren Signale zur Verfügung, um externe Hardware zu bedienen beziehungsweise den HALT-Zustand der CPU aufmerksam zu machen.



High - Bank
Low - Bank
74LS362
U601
TMS 9900
U600
U610
U611
U606
U609
74LS 245
U614
74LS 244
U616
74LS 373
U615
U605
U504
U505
U506
U508
U507
U602
U604
U603
U607
U612
74LS 194
U613
TMS 9901
74 LS 155
U500
U501
U502
GROMϕ
GROM1
GROM2
TIS 92
Q301
Q300
Q402
Q404
Q401
Q403
L400
L401
L402
L403
MAG
1 CASCTR1
2 CASGND1
6 CASCTR2
7 CASGND2
6 MAGOUT
3 MAGOGND
MAGIN
MAGIGND
SOUND-IN
AUDEN
AUDIN
READY-GROM
WEP



Texas Jnstrumentes TI 99-4A Konsole

Hardwarekonzept

Ausgang '138	Bezeichnung	Assemblerbefehl	Wirkung
Q0	/CRUCLK	CRU-Befehle	Synchrontakt
Q1	/IS1	nicht implementiert	–
Q2	/IS2	IDLE (HALT)	Impulsfolge
Q3	/IS3	RESET	Einzelimpuls
Q4	/IS4	nicht implementiert	–
Q5	/IS5	XON	Einzelimpuls
Q6	/IS6	XOFF	Einzelimpuls
Q7	/IS7	LREX	Einzelimpuls

Wenn der Befehl IDLE ausgeführt wird, taktet der Ausgang /IS2 im Rhythmus von fortwährenden CRU-Operationen (proportional CRUCLK). Beendet wird der Befehl durch die Annahme eines Interrupts. Eine LED an diesem Ausgang kann über die Leuchtstärke eine Aussage über die Zeit geben, die der Prozessor im HALT-Zustand verharrt.

2.2 INSTALLATION EINER 16-BIT-ORGANISIERTEN SPEICHERERWEITERUNG

Die Ausführung der Speicherzugriffe auf die 32-KByte-Speicherweiterung läßt sich auf ein Drittel der beim TI-Konzept üblichen Ausführungszeit verringern, wenn der Speicher nicht extern, sondern intern 16 Bit breit analog dem ROM und Scratch-Pad-RAM angelegt wird.
Es werden CMOS-RAM's 6264 verwendet, die, ohne den internen BUS zu belasten, direkt an den Datenbus der CPU angeschlossen werden können. Das erfolgt in zwei Bänken zu je zwei Schaltkreisen. Die Adreß-Anschlüsse A0 . . . A11 der RAMs werden direkt mit den Adressen A1 . . . A12 der CPU verbunden.
Die IS 504 in der Konsole (74LS138) stellt die vier notwendigen Selektsignale /EXL, /EXH1, /EXH2 und /EXH3 für die Selektion der Speichererweiterungs-Bereiche zur Verfügung. Da diese Signale immer 8-KByte-Segmente repräsentieren, entsprechen sie im 16 Bit breiten Speicher je zwei 4-KByte-Blöcken. Beim TMS 9900 sind alle Speicher-Transfers Wort-Transfers, Byte-Transfers setzen sich aus zwei Wort-Transfers zusammen.

Speicheraufteilung der 32-KByte-Erweiterung:

Definitionen:
16 Bit breite Speicheranordnungen unterteilt man in zwei 8 Bit breite Bänke, die man als

- Low-Bank für die Datenbit D0 . . . D7 sowie
- High-Bank für die Datenbit D8 . . . D15 bezeichnet.

Da alle Speicherzugriffe 16 Bit breit erfolgen, heißen die entsprechenden CS-Signale „CS X. ." zur Kennzeichnung dafür, daß beide Bänke selektiert werden.

Syntaxbeispiel: LBL —>
LB: Low-Bank (D0 . . . D7)
L: Low-Teil (Schaltkreis 1)
Low-Teil (Schaltkreis 1)

Zugriff auf >2000 . . . >3FF
untere 4 K von LBL und HBL
—> XBL
Zugriff auf >A000 . . . >BFFF
obere 4 K von LBL und HBL
—> XBL
Zugriff auf >C000 . . . >DFFF
untere 4 K von HBL und HBH
—> XBH
Zugriff auf >E000 . . . >FFFF
obere 4 K von HBL und HBH
—> XBH

Daher wird das Signal /EXL beziehungsweise /EXH2 als Adreßbit A12 zur Selektion des unteren oder oberen 4-KByte-Segmentes für die RAMs der Low- beziehungsweise High-Bank verwendet. Die Segmente /EXL und /EXH1 belegen jeweils einen Schaltkreis für Low- und High-Bank, ebenso die Segmente /EXH2 und /EXH3. Die notwendigen Select-Signale werden mit den Gattern C2 und C3 als high-aktive Signale erzeugt und den high-aktiven CS-Eingängen (Pin 26) der 6264-ICs zugeführt.

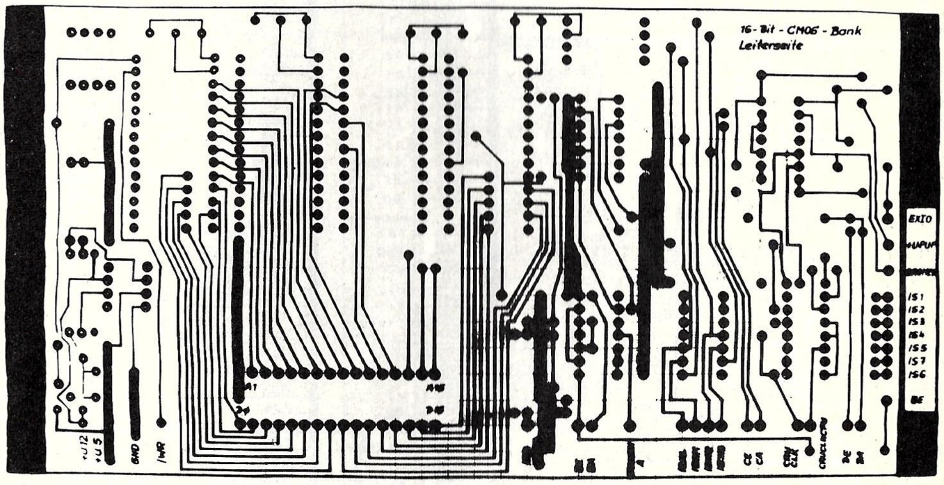


Abb. 4b

Die 32-KByte-Speichererweiterung teilen sich auf in je 16 KByte für die Low- und die High-Bank. Da zur Realisierung 8 KByte-RAM vom Typ 6264 verwendet werden, setzt sich jede Bank noch aus einer internen Low- und High-Bank für die ersten und die letzten 8 KByte zusammen:

Die Widerstands-Kombination dienen der Leitungsanpassung.
Damit die Speicher intern 16 Bit breit angeschlossen werden können, muß der Start der Ablaufsteuerung verhin-

dert werden. Das erreicht man dadurch, daß die Zugriffe auf die Speichererweiterung in gleicher Weise erkannt werden wie die Zugriffe auf den Betriebssystem-ROM und den Scratch-Pad-RAM. Vergleichen Sie dazu die Erläuterungen aus Beitrag 1, „Anatomie".

Das Ausgangssignal des OR-Gatters U 507.3 (Pin 8) wird sowohl von den /CS-Eingängen der internen RAMs (Unterbrechung A in Abbildung 1) als auch vom Pin 12 des NAND-Gatters U 606.4 getrennt (Unterbrechung B in Abbildung 1) und an den Eingang BE der Zusatzplatine übergeben. Zur weiteren externen Verwendung wird das Signal BE auch auf den „größeren" indirekten Steckverbinder geführt.
Das RAM-/CS-Signal wird exakter ausdecodiert. Statt des bisherigen Bereiches >8000 . . . >83FF wird der RAM-Adreßbereich, der eigentlich >8300 . . . >83FF belegt, durch Verknüpfung mit Adreßbit A9 auf den Bereich >8200 . . . >83FF eingeschränkt. Dazu dienen die Gatter A4, A3 und A2:

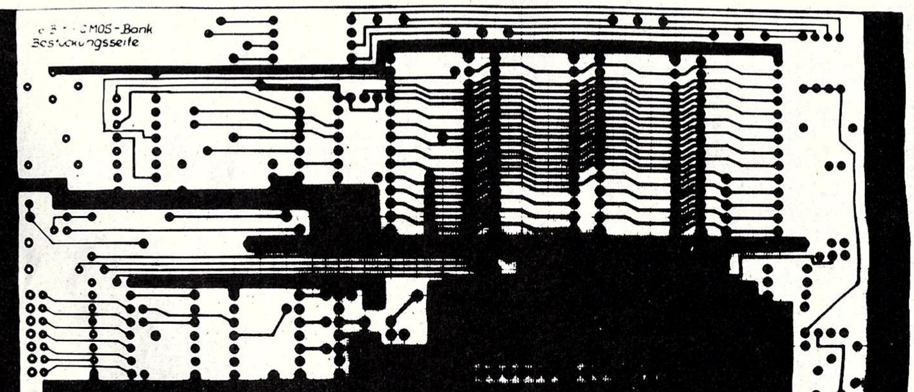


Abb. 4a

Vor- und Rückseite

/RAMEN = /(A9 & /BE) mit BE = (8000 . . . 83FF)

Gleichzeitig steuert das Signal RAMEN am Ausgang des Gatters A3 das NOR-Gatter A1 an und erzeugt das Abschaltsignal für die Ablaufsteuerung /(RAMEN v EXIOEN). Dieses verläßt als Signal BA die Zusatzplatine und wird an den Eingang Pin 12 der IS U 606.4 geführt. Wenn dieses Signal low-aktiv ist, bleibt die Ablaufsteuerung inaktiv. Der Speicherzugriff ist 16 Bit breit und nur zwei T-Zyklen lang, sofern kein zusätzlicher WAIT-Zyklus über den Eingang READY erzwungen wird.

SPEICHERAUFTEILUNG DER 32-KBYTE-ERWEITERUNG

Zur Sicherung der Daten in der Speichererweiterung nach Abschalten der Konsole dient eine Pufferspannung von 3 Volt, die im Mustergerät nur über den indirekten Steckverbinder von außen zuführbar ist. Es ist aber auch der Einbau eines Lithium-Akkus in die Konsole denkbar.
Zur Datensicherung ist eine Zusatzbeschaltung vorgesehen:

Die CMOS-Spannung +UCMOS wird im Normalfall über einen voll durchgesteuerten Transistor T1 aus der geregelten 5-Volt-Spannung der Konsole bereitgestellt, weil das 12-Volt-Netzteil schon arg strapaziert ist. Die Stromaufnahme im Falle der Selektion zweier CMOS-RAMs beträgt etwa 50 bis 60 mA, im nicht selektierten Zustand nur Mikroampere.
Die Basis von T1 wird über einen Vorwiderstand aus +U12 der Konsole gespeist. Der 12-Volt-Spannung der Konsole wird aus einem Festspannungsregler versorgt, die 5-Volt-Spannung dagegen aus einem Schaltnetzteil. Dieses schaltet sofort ab, wenn die 12-Volt-Spannung am Zusammenbrechen ist.
Aus der 12-Volt-Spannung wird mit einer Z-Diode eine Referenzspannung von etwa 5,6 Volt erzeugt, die mit einem Spannungsteiler auf zirka 4,5 Volt geteilt wird. Ein OPV '081 vergleicht diese Referenzspannung mit der tatsächlichen 5-Volt-Spannung. Solange die 5-Voltspannung größer ist als die Referenzspannung von etwa 4,5 Volt, ist der Ausgang des OPV auf High-Potential. Dieses wird auf TTL-Pegel heruntergeteilt. Das Signal wird auf Pin 5 des Triggergatters D2 ('4093) gegeben.
Solange dieser Eingang und Eingang Pin 6, der direkt mit +U5 verbunden ist, auf High-Potential liegen, ist der Ausgang Pin 4 und damit auch Pin 11 low-aktiv. Dieses Signal wird mit /REN (/RAM-Enable) bezeichnet und den low-aktiven Chip-Selekt-Eingängen der 6264-RAMs zugeführt.
Unterschreitet die 5-Volt-Spannung die Referenz von 4,5 Volt, wird das Signal /REN high-aktiv und disabled die RAMs. Um eine versehentliche Freigabe durch Schwingungen des OPV beim Zusammenbruch der 12-Volt-Spannung zu vermeiden, wird auch die 5-Volt-Spannung an dem Gatter D2 mitbewertet.

FÜNF VOLT GARANTIERT DATENSICHERHEIT

Da die 5-Volt-Spannung viel früher verschwindet als die 12-Volt-Spannung, ist eine ordentliche Sicherung der Daten der RAMs gewährleistet.
Im Pufferungsfall werden die RAMs und der Triggerschaltkreis, über eine Diode entkoppelt, aus der Spannungsquelle +UPUF versorgt. Eine weitere Diode sichert, daß die CMOS-Spannung keinesfalls mehr als 0,6 Volt unter die 5-Volt-Spannung fällt.
Schaltungstechnisch ist dies allerdings kaum möglich.

Heinrich Cronjaeger

DISKETTENLAUFWERKE

ALL SYSTEM GO!

Zu Diskettenanwendungen im Software-Bereich ist in diesem Magazin bereits einiges gesagt worden. Doch am anderen, dem Hardware-Ufer, wurde bis jetzt noch nicht angelegt. Bis jetzt.

Wer behaupten würde, die Diskette sei nicht das Speichermedium überhaupt geworden, könnte zu Recht nicht ganz ernst genommen werden. Da gelten ähnliche Argumente wie bei einer alten Mercedes-300 SEL-6.3 Liter-Limousine: Es gibt zwar viel Besseres und Schnelleres, aber noch viel, viel Schlechteres und Langsameres.

Die Art, wie die Einbindung des DISK-EXPANSION-SYSTEM am TI vorgenommen wurde, ist heute noch Stand der Dinge und wird es auch prinzipiell bleiben. Gemeint ist, das System in die Einzelkomponenten

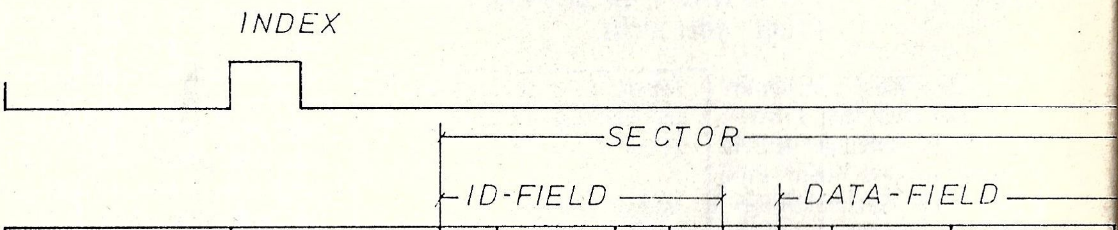
– Diskettenlaufwerk(e),
– Diskettenlaufwerkkontroller und
– Steuersoftware

zu zerlegen. Damit kann ein Teil des Systems gegen ein anderes, weil besseres, ausgetauscht werden, ohne alles andere einzumotten. Das gilt grundsätzlich für alle Computer, die die Diskettenantriebe wie der TI-99/4A einbinden. Das Prinzip der DSR schafft zudem die Möglichekeit, den Computer auch ohne Disk-System zu betreiben. Er ist also aufwärtskompatibel. Ein IBM-PC unter dem Betriebssystem MS-DOS ist ohne Laufwerke ein Dekorationsobjekt.

Als TI den Rechner 99/4A entwarf, griff man zu einem Laufwerksformat, welches zu diesem Zeitpunkt in der Peripherielandschaft preiswert greifbar war. Es erlaubt es, die 5.25"-Diskette auf einer Seite, mit einfacher Dichte, auf 40 Sputen, softsektoriert zu beschreiben.

Softsektoriert bedeutet, daß die verwendeten Disketten nur ein Indexloch haben. Dieses Loch definiert den Zählbeginn des Kontrollers zum Wiederfinden der Sektoren.

Stellen Sie sich eine Diskette im Laufwerk (Drive) vor. Es handelt sich, ganz grob, um eine flexible Scheibe, mit einem großen Loch in der Mitte und einem vergleichsweise kleinen Indexloch in der Diskette, welches sich 25 Millimeter von der gedachten Diskettenmitte entfernt befindet.



	G1	S	AM1	ID	CR	G2	S	AM2	DATA
FM	40	6	1	4	2	11	6	1	128
MFM	80	12	3 1	4	2	22	12	3 1	256

Beim Betrieb des Laufwerks greift sich eine Achse des Antriebsmotors, über Riemen oder ähnliches übersetzt, die Diskette. Diese macht die Drehungen des Motors mit, während eine Anordnung von Fotobauteilen das Auftreten des Indexloches erfaßt. Alle Laufwerke drehen die Diskette mit 300 Upm (verbindlicher Wert). Ohne höhere Mathematik erkennt man schnell, daß dies 5 Ups (Umdrehungen pro Sekunde) sind. Um die Diskette einmal um ihre eigene Achse, also um 360 Grad, zu drehen, vergeht daher eine Zeit von 0.20 Sekunden.

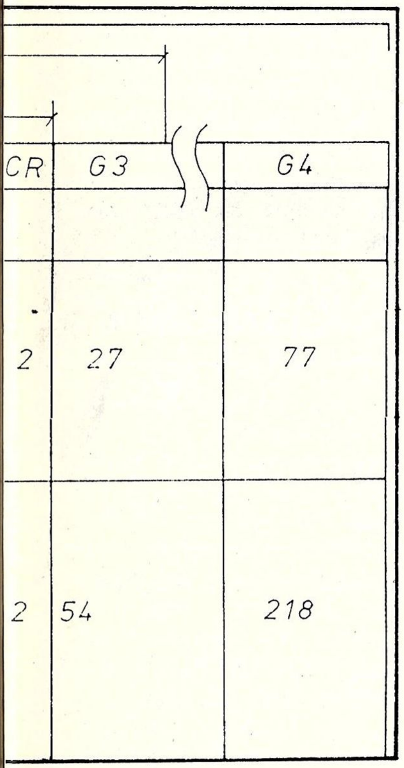
Daraus ergibt sich, daß die Fotobauteile im Verbund mit den anderen Teilen der Laufwerks-Steuerelektronik alle 0.2 Sekunden einen Indeximpuls abgeben. Innerhalb dieser Zeit überstreicht der Schreib-/Lesekopf des Laufwerks die komplette Diskettenspur und damit alle ihrer Sektoren. Das können theoretisch nahezu beliebig viele sein. Ihre Anzahl wird bei der Wahl des Datenformats einmalig festgelegt und bleibt dann verbindlich.

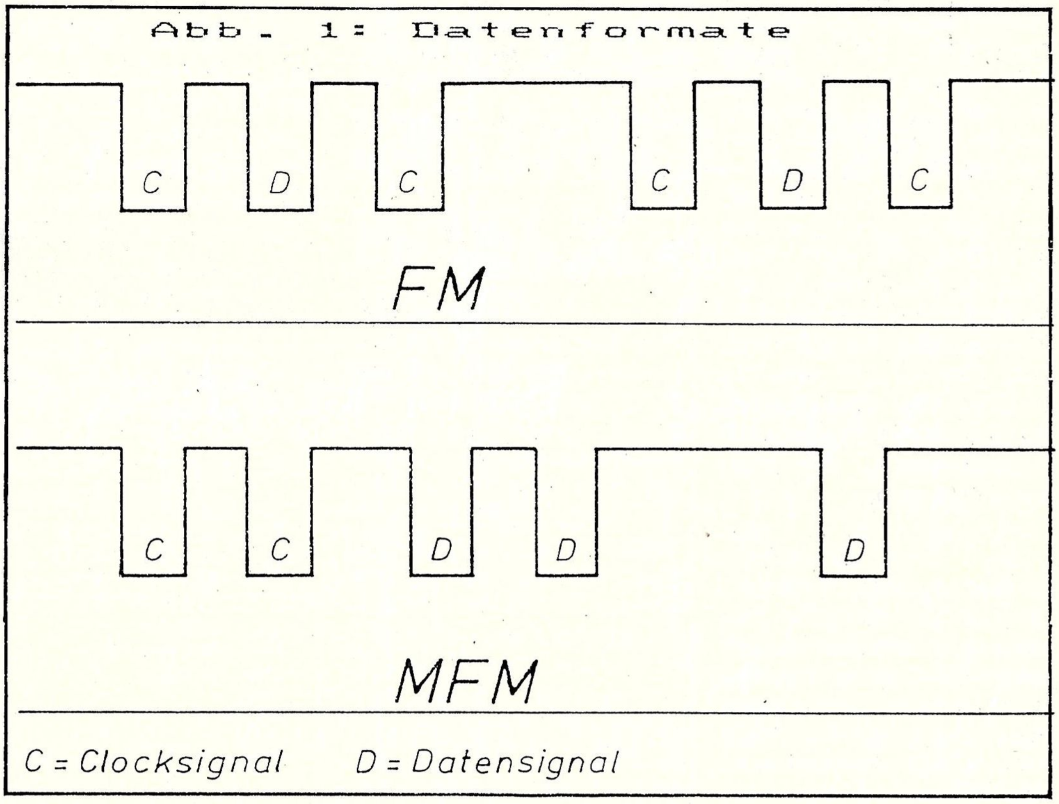
Das soll nicht heißen, daß man es nicht über Software ändern könnte. Im Falle des TI-Diskettensystems wurde ein Wert von neun Sektoren pro Spur festgelegt. Wenn der Schreib-/Lesekopf in 0.2 Sekunden einmal über die Spur geführt wird, streicht er also an genau neun Sekunden vorbei.

Um einen dieser Sektoren gezielt anfahren zu können, muß man nur wissen, wieviel Zeit seit dem Auftauchen des Indexloches vergangen ist. Bei 0.20 Sekunden für alle neun Sektoren einer Spur dauert es 0.2s/9 (gleich 0.02222s), um einen einzelnen Sektor weiterzuschalten. Will der Controller zum Beispiel den Sektor Nummer 3 auf einer bestimmten Spur anfahren, wartet er die Dauer von 2x(0.2s/9) ab. In dieser Zeit passiert der Schreib-/Lesekopf den ersten und zweiten Sektor und befindet sich dann am Anfang des dritten, der für die Dauer von 0.022222s gelesen beziehungsweise beschrieben werden kann.

Für die einwandfreie Funktion dieses etwas kom-

plexen Vorgangs, ist die Drehzahl des Laufwerksmotors (hier: 300 Upm) beziehungsweise deren Konstanz eine Grundvoraussetzung. All das hat mit dem Laufwerk selbst nicht weiter zu tun. Vielmehr werden alle Daten vom Controller so erwartet, und Veränderungen wären theoretisch denkbar.





Beim TI-Original-Kontroller arbeitet man mit 40 Spuren (englisch: tracks), neun Sektoren pro Spur und somit 360 Sektoren pro Diskettenseite. Zieht man die für den Diskettenkatalog vom System belegten zwei Sektoren ab, erhält man bei einer leeren, initialisierten Diskette die bekannte Meldung: 358 free. Wer an der softwaremäßigen Nutzung der Diskette beim TI-99/4A näher interessiert ist, möge in der TI-REVUE Ausgabe 6/1986 nachschlagen.

Ein letzter Software-Ausflug in diesem Zusammenhang: Durch die softsektorierte Handhabung kann man mit ein und derselben Diskette verschiedenes anfangen. Bei jeweils einfacher Dichte gibt's zum Beispiel bei 16 Sektoren à 128 Byte 81,9 KByte, bei fünf Sektoren à 512 Byte 102,4 KByte. Beim TI-Diskcontroller und baugleichen, in Verbindung mit der Software des DISK-MANAGER II oder des DM 1000 ergeben neun Sektoren à 256 Byte 92,2 KByte.

KByte. All das bei einer Diskette, die unformatiert auf 40 Spuren, einseitig und einfach dicht immer 125 KByte faßt.

So erklären sich auch die vielen formatierten Datenmengen und verschiedenen Formatsbezeichnungen wie ISO-Standard, IBM-Standard oder OS-Standard (Ohio Scientific OS-65D Operating System).

Bei hardsektorierten Disketten befindet sich vor jeder Sektorenkerbe (beim TI wären dies 9) ein Indexloch. Dies bindet aber an das durch die Zahl der Indexlöcher vorgegebene Format. Alle Kontroller arbeiten beim TI softsektoriert und daher auch alle Laufwerke mit den Disketten. Grundsätzlich kann ein Laufwerk aber mit beiden Möglichkeiten etwas anfangen, die weiterverarbeitende Hard- und Software aber nicht unbedingt.

Die nächste Steigerung der speicherbaren Datenmenge ergibt sich, indem man die Aufzeichnungsdichte verdoppelt. Dieser Vorgang wird unter dem Kürzel DD (Double Density) gehandelt, im Gegensatz zu SD (Single Density), dem Format mit einfacher Dichte. Man unterscheidet SD und DD technisch gesehen am Aufzeichnungsformat: FM für SD und MFM für DD. Abbildung 1 zeigt den Unterschied auf: So sehen die Datenformate aus, die der Schreib-/Lesekopf des Laufwerks auf die Scheibe bringt. Gezeigt werden hier nur die ersten Zyklen der Datenformate.

Das Diagramm zeigt am 16-Sektoren/Spur-Beispiel den genauen Aufbau eines Sektors. Letztlich führt der Umstieg auf die doppelte Dichte zu einer Verdopplung der speicherbaren formatierten Kapazität.

Die Daten werden vom Controller zum Laufwerk, und umgekehrt, seriell übertragen. Die Signale haben aber sowohl elektrisch als auch mechanisch ein ganz anderes Protokoll als etwa die serielle V24/RS232C-Schnittstelle: Die vom Rechnerbus kommenden parallelen Daten werden im Controller in serielle verwandelt, mit Zeittakten gemischt (moduliert) und zum Laufwerk weitergegeben. Beim Lesen trennt der Controller diese Zeitsignale (Clock-Impulse) wieder von den Daten ab und gibt diese parallel an den Rechnerbus, zu dessen Verwendung, weiter.

Die Geschwindigkeiten dieser Schnittstellen liegen bei 125 KBit/s für FM und 250 KBit/s für MFM. Um die aufzeichenbare Datenmenge weiter zu erhöhen, ist der nächste Schritt die Verdopplung des Schreib-/Lesekopfes: Der zweite Kopf bedient die zweite Diskettenseite und ver-

doppelt so die Datenmenge erneut.

Die untere Seite der Diskette ist immer Seite 1, die obere, die mit dem Label, immer Seite 2. Spur 1 wird immer als Spur 00, die 40ste immer als Spur 39 bezeichnet. Auf der ersten Diskettenseite wird am äußeren Diskettenrand mit dem Zählen der Spuren begonnen, auf der zweiten dagegen am inneren Rand. Spur 3 auf der ersten Seite ist also die vierte Spur von außen, Spur 3 auf der zweiten Seite die vierte Spur von innen.

Die Verdopplung führt zu den einschlägig bekannten ATRONIK-, MYARC- und CorComp-Controllern, die alle doppelseitig und doppeltdicht arbeiten können (DS/DD). Der Urvater, der TI-Diskkontroller, kann, bis auf die doppelte Dichte und die Beschränkung auf drei Laufwerke, dies auch.

Trotz dieser Unterschiede ist aber alles aufwärtskompatibel, nur der Kontroller bestimmt die maximale Datenmenge. Die Laufwerke akzeptieren ohnehin alles. Wenn eine Diskette mit den Attributen SS/SD initialisiert wurde, kann sie der Controller mit DS/DD problemlos lesen. Schließlich verlangt ihm dies nur ein Viertel seines Könnens ab.

Wie bei NF-Übertragungssystemen auf analoger Basis ist auch hier die Gesamtheit nur so gut wie das schlechteste Einzelteil. Ein DS/DD-Controller kann nur dann DS bedienen, wenn das Laufwerk zwei Köpfe hat. Zur Verdeutlichung siehe die Tabelle in *Abbildung 2*. Sie zeigt, was man kombinieren kann und was es bringt.

Vor Jahren wurden noch Laufwerke hergestellt, die einen oder zwei Köpfe hatten, aber die doppelte Dichte nicht verarbeiten konnten. Die meisten Original-TI-Laufwerke sind aus diesen Serien. Heute wird etwas derartiges nicht mehr neu gefertigt. Insbesondere die aktuellen Laufwerke mit nur der halben Bauhöhe, geringer Stromaufnahme und schnelleren Zugriffs- und Positionierungszeiten (unter dem Begriff Slimline bekannt) sind mit Sicherheit für mindestens doppelte Dichte gut.

Der Hersteller spielt dabei keine Rolle, da keiner schlechter sein möchte als die Konkurrenz. Es gibt nur qualitative Unterschiede, etwa in der Lebensdauer. Jedoch besteht die Möglichkeit, Restposten oder aus Redundanzgründen ausgewechselte Laufwerke für einen Bruchteil des Listenpreises zu erstehen. Wenn der Preis günstig ist, kann man getrost auf Halde kaufen.

Wer viel mit einschlägiger Software wie MULTIPLAN oder E/A arbeitet, ist wahrscheinlich kurz vor einem Anfall, wenn er nur ein Diskettenlaufwerk besitzt. Der ständige Wechsel von System- und Datendiskette raubt einem fast den Verstand. Mindestens zwei Laufwerke, eventuell einseitig und mit einfacher Dichte, sind daher in fast allen Fällen einem Non-Plus-Ultra-Laufwerk vorzuziehen. Man denke nur an den Wunsch nach einer Sicherungskopie (BACKUP) bei nur einem Laufwerk.

„An meinen C64 kann ich ja auch nur die 1541 anschließen." Schon klar. Aber TI hat das eigentliche Diskettensystem dreigeteilt und sehr wichtig, dabei Komponenten verwendet, die nicht exklusiv von und für TI gebacken wurden.

Bei der 1541 handelt es sich zum Beispiel um eine in sich geschlossene Einheit mit eigenem Betriebssystem, das ebenfalls nur in sich geschlossen arbeitet. Will man zwei Laufwerke, muß man das gesamte System verdoppeln, eben eine zweite 1541 anschaffen. Das ist unsinnig, besonders wenn man weiß, daß die 1541 nur mit einem „Tempo" von 300 Baud mit dem 64er kommuniziert. Beim 99er brauche ich nur einen zweiten Diskettenantrieb. Naturgemäß ist daher die TI-Anlage mit nur einem Laufwerk recht teuer.

Abb. 2 Datenmengen

mögliche verwendete Kontroller	mögliche Laufwerke	mögliches verwendbares Format bei INIT	maximal erzielbare Speichermenge	maximale Anzahl der Laufwerke
TI-Original-Controller	40 SS/SD	40/SS/SD	ca. 90 KByte	3
	40 SS/DD	40/SS/SD	ca. 90 KByte	3
	40 DS/SD	40/SS/SD	ca. 90 KByte	3
		40/DS/SD	ca. 180 KByte	3
	40 DS/DD	40/SS/SD	ca. 90 KByte	3
		40/DS/SD	ca. 180 KByte	3
Nachfolger des TI-Controllers von MYARC, ATRONIK, CorCom....	40 SS/SD	40/SS/SD	ca. 90 KByte	4
	40 SS/DD	40/SS/SD	ca. 90 KByte	4
		40/SS/DD	ca. 180 KByte	4
	40 DS/SD	40/SS/SD	ca. 90 KByte	4
		40/DS/SD	ca. 180 KByte	4
	40 DS/DD	40/SS/SD	ca. 90 KByte	4
		40/DS/SD	ca. 180 KByte	4
		40/SS/DD	ca. 180 KByte	4
		40/DS/DD	ca. 360 KByte	4

Dabei bedeutet:
SD = Single Density (Einfache Dichte),
DD = Double Density (Doppelte Dichte),
SS = Single Sided (Einseitig) und
DS = Double Sided (Zweiseitig).

Die Kombination Controller-Diskettenantrieb ist deshalb so variabel, weil beide, Controller und Diskettenantrieb, mit einer Schnittstelle arbeiten, die, ähnlich wie die CENTRONICS-8Bit-Parallel-Schnittstelle, eine sogenannte Quasinorm darstellt. Diese Schnittstelle wurde von der Firma SHUGART ersonnen, einem Betrieb, den es leider in seiner ursprünglichen Form nicht mehr gibt.

Der Schnittstellenbus bekam in Anlehnung an den Firmennamen die legendäre Bezeichnung SHUGART-BUS. Das magische Wort, wonach der potentielle Käufer Ausschau halten muß, ist daher „shugartbus-kompatibel".

BRÜCKEN BAUEN

Zwar gibt es Hersteller, die diesen Bus nach ihren Vorstellungen modifiziert haben, etwa BASF, jedoch ist mir kein Fall bekannt, bei dem nicht durch Umstecken einiger Brücken auf der Platine des Laufwerks sogenannte Jumper die Kompatibilität zum SHUGART-BUS wiederhergestellt werden kann.

Eine Ausnahme der Regel: Die auch von SHUGART gebauten 8"-Laufwerke haben einen aufgrund des Formats aufwendigeren Bus (50polig). Dieser ist elektrisch kompatibel zum 5.25"-Bus, man muß aber einen Adapter einsetzen. Für die 5.25"-Drives ist der 34polige Bus die Norm. Diesen Bus kann man auch, zumindest elektrisch, an 3"- und 3.5"-Laufwerken finden. Daher steht auch dem Einsatz dieser Diskettenformate nichts mehr im Wege, es gelten die gleichen Auswahlkriterien wie bei den 5.25"-Typen.

Damit man die Zusammenhänge wirklich beurteilen kann, muß man den Bus genau kennen. Nicht nur ihn werden ich in der nächsten Ausgabe vorstellen.

J. Brilmayer

DIE GLÄSERNE CPU

Gelegentlich wurde und wird uns vorgehalten, unser TI sei eine kreuzlahme Krücke, der man beim Rechnen zusehen könne. In den zurückliegenden Jahren hat diese unqualifizierte Äußerung von TI-Fremden für manchen Frust gesorgt. Doch inzwischen sind aufgeklärte Zeiten angebrochen, und jeder ist mittlerweile in der Lage, sofort das korrekte Argument zurückzuschmettern: „Schließlich arbeitet der TI bereit assemblernah mit einer Art Hochsprache, die interpretiert wird. Und damit ist er im Prinzip doch der schnellste!"

Einem sehr talentierten Programmierer namens D.C. Warren war das immer noch nicht genug. Er meinte, daß man auch den Maschinencode noch interpretieren müsse. Damit würde all das, was die CPU macht, beobachtbar und manipulierbar werden. Was dabei heraus kam, war besagter Explorer, vertrieben von Millers Graphics, geschrieben 1985.

Doch nicht allein das Programm ist es, das für Aufsehen sorgt. Auch die Anleitung ist eine Klasse für sich. Sie umfaßt sage und schreibe 105 Seiten, prall gefüllt mit Informationen zum Software-Innenleben des TI-99/4A, die in dieser Kompaktheit und derart umfassend in keiner anderen Publikation zu finden sind.

Hier werden zum Beispiel die diversen weißen Flecke im TI-99/4A – Intern von Heiner Martin erklärt – ein paar Datas waren wohl doch Op-Codes. Man findet sehr übersichtliche Tafeln zur Speicheraufteilung des 4A generell sowie zu den Blöcken der diversen Programmteile im Konsolen-ROM und -GROM. Immer noch neu ist die ausführliche Beschreibung der Benutzung von PAD-RAM, Low- und High-Memory sowie VDP-RAM durch Extended BASIC. Das dürfte zu einer Flut von neuen Peeks und Loads für den Abschuß des BASIC-Interpreters führen.

Daß diese vielen Segmente nicht in aller Ausführlichkeit abgehandelt werden konnten, liegt auf der Hand. Dem Autor des Manuals, Craig G. Miller, ist es jedoch gelungen, Information und Umfang befriedigend gegeneinander anzuwägen.

HOHES LADE-TEMPO

Beim Programm überrascht zuerst, daß der Extended-BASIC-Autostart fehlt – es ist kein LOAD auf der Diskette. Also lädt man aus dem EA-Modul das File EXP. Die zweite Überraschung ist das extrem hohe Ladetempo, bei dem man glauben könnte, das Laufwerk sei kaputt. Ist es aber nicht – auch dann nicht, wenn sich auf einem CPS-99 gar nichts tut.

Hat man das Titelbild mit der Copyright-Meldung und dem grafisch einfach gehaltenen, aber trotzdem professionell aussehenden Programmnamen durch Druck auf irgendeine Taste verscheucht, so wird man mit einer wahren Informationsflut konfrontiert. Hier hilft nur systematisches Vorgehen.

EXPLORER verfügt über drei verschiedene Arbeitsschirme. Davon ist der erste, den man nach dem Start sieht, der Haupt- oder Kontrollschirm, von dem aus alle Abläufe gesteuert und überwacht werden. Fast alle Zahlentasten sind zusammen mit FCIN oder CTRL definiert. Der Hauptschirm kann in fünf Zonen unterschiedlicher Bedeutung aufgeteilt werden: Ganz oben findet sich die Protokollierung der Zustände von CPU-, VDP- und GRAM beziehungsweise GROM sowie die Bit-Interpretation des Statusregisters der CPU. Ganz links wird angezeigt, welcher Workspace-Pointer momentan geladen ist, an welcher Speicheradresse die CPU gerade arbeitet und welchen Zustand das Statusregister hexadezimal hat (bitweise steht das oben rechts).

Analog hierzu sind die Anzeigen von VDP-RAM und GROM beziehungsweise GRAM aufgebaut. 'ad' gibt die aktuelle Adresse des internen Adreßzählers an, 'by' das zuletzt hieraus gelesene Byte. 'st' ist das Statusregister beim VDP. Da es das beim GROM nicht gibt, steht an dieser Stelle das GPL-Statusbyte.

'bp' ist etwas ganz feines: Hiermit können Breakpoints gesetzt werden, und zwar entweder im CPU-RAM/ROM, VDP-RAM oder GROM/GRAM.

Damit kann der Programmablauf immer dann gestoppt werden, wenn einer dieser Speicher diese Adresse erreicht. Beim VDP gibt es nette Variationen über die Steuermasken 40 und 80, wodurch zum Beispiel, bei einem Zugriff auf ein bestimmtes VDP-Register, eine Unterbrechung ausgelöst werden kann.

Explizit abschalten kann man die Breakpoints nicht, aber wenn man sie auf FFFF setzt, treten so gut wie nie Unterbrechungen auf.

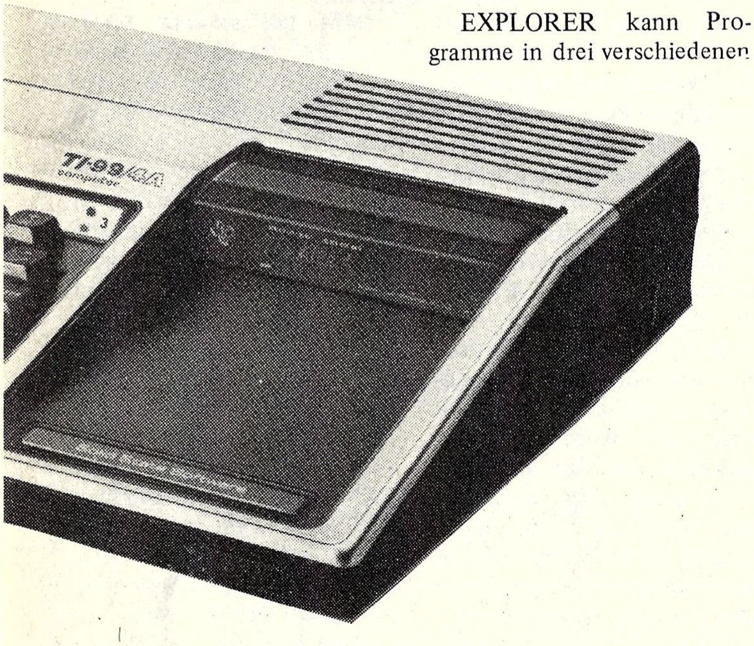
Die zweite Anzeigezone darunter gibt den Inhalt aller Register im momentan geladenen Workspace an. Somit kann man beobachten, wie welcher Befehl welches Register beeinflußt.

Unter dieser Zone befindet sich die Anzeige für die Inhalte der acht VDP-Register. Sie dient nur dem schnellen Überblick. Ausführlicher zeigt dies ein anderer Schirm.

Anzeigezone vier schließlich ist das direkte Fenster in einen der drei Speicherbe-

reiche des TI. Begrenzt ist es mit dem '=' -ASCII-Symbol. Obenauf sitzen drei Zusatzinformationen, aus denen entnommen werden kann, welcher Speichertyp angezeigt wird, welche Adresse das erste Zeichen der zweiten Zeile des Fensters hat (sehr praktisch, da so sichtbar ist, was vorher kam und was noch folgt), welche GROM-Basis aktuell ist und wie viele Befehle bisher abgearbeitet wurden (Instruktonszähler, ganz rechts).

Rechts neben der Adresse des Speicherfensters steht die Betriebsart des Fensters. 's' bedeutet, daß sich der Inhalt nur nach Tastendruck auf dem Hauptschirm (FCIN 4



oder 6) weiterbewegt. Bei 'd' ändert sich die Adresse des Fensters dynamisch, je nachdem, wo sich das Programm, das simuliert wird, im gewählten Speicherblock bewegt.

Zuunterst findet sich eine ein- oder dreizeilige umschaltbare Anzeige. In ihr wird der disassemblierte Inhalt des Speicherbereichs angezeigt, an dem sich der PC der CPU gerade befindet (sinnigerweise nur CPU-RAM/ROM). Gemäß der Definition des PC (Programm-Schritt-Zähler, Programm Counter) ist das der als nächstes auszuführende Befehl. Dieser Kniff erlaubt es, einen Befehl zu sehen, bevor er abgearbeitet wird. Man kann sich also vor der Ausführung Gedanken darüber machen, was passieren wird und nachher prüfen, ob die Überlegung stimmte.

Man kann auf dem Hauptschirm mit den Cursortasten (FCTN E,S,D,X) beliebig „spazierenfahren" und nach Lust und Laune Inhalte von Registern, den PC oder den Workspace-Pointer manipulieren. Natürlich nicht ganz frei, denn das simulierte Programm muß ja nicht unbedingt abgeschossen werden.

EXPLORER kann Programme in drei verschiedenen Modi ablaufen lassen. Dabei kann es sich um Programm handeln, das zuvor, vor EXPLORER, geladen wurde. Es können auch Teile des Konsolen-Betriebssystems sein, wenn man sich etwa alle Vorgänge nach dem Einschalten des Rechners besehen will (Power Up).

Es gibt den Einzelschritt-Modus, der mit der Tastenkombination CTRL 1 ausgelöst wird. Dabei wird pro Tastendruck nur ein Befehl abgearbeitet, die Taste hat keine Wiederholfunktion. Es handelt sich also um einen Single-Stepper, der ohne zusätzliche Hardware auskommt.

Die zweite Betriebsart ist Dauerschritt-Modus (CTRL 2). Dabei arbeitet EXPLORER einen Befehl ab, bringt den Schirm auf den neuen Stand und befaßt sich mit dem nächsten Befehl. Er tut dies so lange, bis erneut CTRL 2 gedrückt wird.

Wenn man sich den ständigen Neuaufbau des Bildschirms sparen will, muß man bei laufendem Programm nur die Shift-Taste drücken und festhalten. Das ist die dritte Betriebsart, der Turbo-Modus, wo EXPLORER sich den Bildschirmaufbau spart und nur die oberste Anzeigezone erneuert. Das beschleunigt die Sache enorm, zum Beispiel können Schleifen schneller durchlaufen werden. Drückt man dazu die Leer-, die ENTER- oder CTRL-Taste, läuft auch der Instruktionszähler mit.

FRAGE ....

Aber was hilft der ganze Aufwand, wenn man nicht einmal sieht, was das Programm im Endeffekt macht? Die Antwort liefert ein Druck auf CTRL 3. All das, was das von EXPLORER simulierte Programm macht, geschieht gewissermaßen auf einem Hintergrund-Bildschirm, der bei Bedarf mit CTRL 3 zu- oder abgeschaltet werden kann.

Mit CTRL 3 kann man also wählen, ob man das simulierte Programm beobachten will oder sehen will, was die CPU gerade macht. Bei eingeschaltetem Hintergrundschirm sind die Tasten CTRL 1 und 2 aktiv. Zudem laufen die Programme noch schneller ab, da sich EXPLORER den ganzen Anzeigekram des Hauptschirmes sparen kann.

In der Tat ist Geschwindigkeit das, was EXPLORER manchmal fehlt. Das eingangs Gesagte trifft natürlich zu, ist doch das Herzstück von EXPLORER eine Software-Emulation der TMS9900-CPU im TI-99/4A. EXPLORER holt sich also immer den Befehl, auf den sein PC momentan zeigt, interpretiert ihn und setzt Bit, ändert Register oder greift allgemein auf Speicherbereiche zu. Es liegt auf der Hand, daß Befehle, die vorher nur wenige Mikrosekunden zur Bearbeitung benötigten, nun im Bereich von Millisekunden bearbeitet werden.

Am wenigsten stört das bei eigenen Assemblerprogrammen, wenn diese von EXPLORER interpretiert werden. EXPLORER benötigt nur den Speicherbereich <B7D8 bis <FFD7, der Rest steht eigenen Programmen zur Verfügung. Die Klimmzüge, die bei der Anwendung von EXPLORER aus Extended BASIC nötig sind, sollen weiter unten aufgeführt werden.

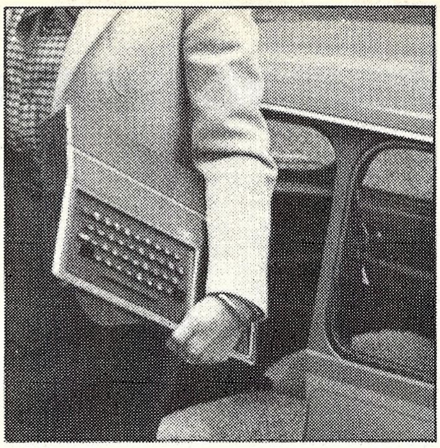
Bei dieser Anwendung wird es ohnehin haarig. Schließlich interpretiert EXPLORER ein Maschinenprogramm, das den GPL-Interpreter darstellt. Dieser arbeitet den BASIC-Interpreter in GPL ab. Der schließlich interpretiert, wie der Name sagt, Ihr BASIC- oder XBASIC-

.... UND ANTWORT

Programm. Nicht nur, daß die Geschwindigkeit langsamer als langsam ist. Viel schlimmer ist, daß etliche Ebenen zwischen den angezeigten CPU-Registern sowie sonstigen Daten und dem eigentlichen Programm liegen. Das verschleiert den Bezug.

Jetzt zu den Funktionen im EXPLORER-Hauptbildschirm:

CTRL 1, 2 und 3 wurden bereits erklärt. CTRL 4 schaltet die Interrupt-Bearbeitung ab. EXPLORER arbeit immer dann diese Routine ab,



wenn der Befehl LIMI 1 oder LIMI 2 gefunden wurde. Das kann mit CTRL 4 an- oder abgeschaltet werden.

CTRL 5 schont das Nervenkostüm all derer, bei denen der Lautstärkeregler am Monitor kaputt ist, denn damit läßt sich der Tongenerator abschalten. Da normalerweise die Dauer eines Tons durch den VDP-Interrupt bestimmt wird, dies bei EXPLORER aber entschieden länger dauert als sonst, werden auch die Töne recht langsam umgeschaltet. Es wird wohl niemand EXPLORER als gebremsten Musik-Computer nutzen, also weg mit dem Gepiepse.

CTRL = ist die gefährlichste Funktion, da sich EXPLORER ohne Sicherheitsabfrage zum TI-Titelbild mogelt.

FCTN 1 schaltet das Speicherbereichs-Fenster durch die drei verschiedenen Speichertypen des TI (CPU, VDP und GROM/GRAM, Anzeige c, v oder g).

FCTN 2 verändert die Größe des Speicherfensters. Es kann fast auf den kompletten Bildschirm ausgedehnt werden, wobei nacheinander die VDP-Register, die Workspace-Register und zum Schluß die oberste Anzeigezone geopfert werden.

FCTN 3 variiert die Größe der Disassembler-Zone zwischen einer und drei Zeilen.

FCTN 4 wirkt auf das Speicherfenster und blättert eine „Seite" nach unten, scrollt also das Speicherfenster um die Anzahl der momentan angezeigten Zeilen weiter zu höheren Adressen hin.

FCTN 6 bewirkt das gleiche zu niedrigeren Adressen hin.

FCTN 5 aktiviert die Suchfunktion (SEARCH).

FCTN 9 bringt den Cursor in das Speicherfenster (bei nochmaligem Druck wieder heraus), wo er frei umherbewegt werden kann. Dieser Editor ist aus verschiedenen Gründen recht langsam: Zum einen gibt sich EXPLORER zugeknöpft, wenn das Speicherfenster Bereiche abdeckt, in denen er residiert. Zum anderen werden Eingaben sofort abgearbeitet und der Fensterinhalt auf den neuen Stand gebracht. Das macht sich besonders in ROM-Bereichen bemerkbar, da der eingegebene Wert zwar kurz sichtbar ist, kurz darauf aber wieder der alte Wert auftaucht. Der RAM-Bereich in der 32-KByte-Erweiterung ist jedoch nicht transparent, da EXPLORER zwischen <B7D8 und <FFD7 vorgibt, hier befinde sich ROM mit dem Wert <FF. Auch den LOAD-Interrupt-Vektor läßt er sich nicht manipulieren.

Der VDP-RAM-Editor ist vollkommen transparent, weil EXPLORER immer den Teil, den seine Screen-Image- und Pattern-Descriptor-Table belegen, dorthin legt, wo das Speicherfenster gerade nicht hinzeigt und den normalen Inhalt in einen Puffer schafft.

FCTN 0 erlaubt es, die Anzeige von ASCII-Symbolen wahlweise mit oder ohne Screen-Offset anzuzeigen (BASIC BIAS). Das wird auch bei der Suchroutine beachtet.

FCTN = dient, wie auch bei Advanced Diagnostics, der Umschaltung von ASCII- und Hex-Darstellung.

DIE SUCHROUTINE

Nach Druck auf FCTN 5 erscheint im Speicherfenster eine Zusatzzeile, in welcher man ein beliebiges Muster eingeben kann. Nach ihm wird dem Typ von Speicher gesucht, den das Speicherfenster momentan anspricht. Dabei hängt es von der Darstellung im Speicherfenster ab (Hexadezimal oder ASCII, schaltbar mit FCTN =), ob ein Bytemuster oder ein String eingegeben werden kann.

Über der Reihe aus '=' verschwindet die GROM-Basis und es erscheinen zwei Felder, welche die Start- ('st', Start) und Endadressen ('fn', finish) für die Suche angegeben. Bei Eingabe eines Strings (ASCII-Modus) wird der eventuell aktivierte BASIC-BIAS beachtet. Dabei wird nach dem Druck auf ENTER mit der Suche begonnen, und zwar nach der Bytefolge, die sich vom linken Rand der Zusatzteile bis zur Cursorposition ergibt. Wurde die Folge gefunden, so wird diese Adresse links oben angezeigt und gleich die neue Startadresse für die Suche auf ein Byte dahinter eingestellt – alles klar zum Weitersuchen.

DER OPTIONS-BILDSCHIRM

FCTN 7 schaltet um auf den Options-Schirm, bei nochmaligem Druck zurück auf den Kontrollschirm. Der Options-Schirm stellt einen Binär-Dezimal-Hexadezimal-Rechner zur Verfügung, der alle logischen Operationen in den drei Zahlensystemen (auch gemischt) durchführen kann.

Durch Druck auf FCTN 9 springt der Cursor in das Operationsfeld, wo er auf die gewünschte Operation gefahren werden kann. Diese wird durch Druck auf ENTER ausgeführt. FCTN 9 führt zurück in den freien Cursorbetrieb.

Unter dem Rechnerfeld befindet sich das eigentliche Optionsfeld. Hier können die Farben eingestellt werden, die EXPLORER bei einer falschen Taste als Warnung zeigt oder mit denen er einen erreichten Unterbrechungspunkt (BREAKPOINT) signalisiert. Daneben wird der dezimale Wert des Instruktionszählers gezeigt, den man manipulieren kann (wie im Hauptschirm): Er läßt sich als schneller HEX-DEZ-Umsetzer in 32 Bit Breite „mißbrauchen".

Mit Cru-Base kann ein 4-Bit-Muster an eine frei wählbare CRU-Adresse geschrieben werden. Dadurch ist es möglich, DSRs zu analysieren. Man sollte jedoch die zeitkritischen Operationen wie in der RS-Schnittstelle oder dem Diskcontroller meiden. Die Schnittstelle nimmt es normalerweise nicht übel, nur funktioniert nichts so richtig. Beim Diskcontroller aber kann man sich durchaus eine Diskette ruinieren.

Abschießen kann man EXPLORER dann, wenn man an seine CorComp- oder Myarc-RAM-Disk heran will. Die einzige RAM-Disk, die mit EXPLORER untersucht werden kann und dabei noch richtig arbeitet, ist die von HORIZON.

Die Optionen „Load Characters" und „Execute Key Scan" erlauben es, daß EXPLORER Ausnahmen macht und bestimmte Programmsegmente schneller ablaufen läßt. „h" an diesen Stellen bedeutet, daß EXPLORER diese reservierten Sequenzen nicht interpretiert, sondern kurzzeitig die Zügel locker läßt. Die erste Option ist in BASIC-Anwendungen nützlich, da beide Interpreter, TI- und Extended BASIC, zwischendurch den Zeichensatz nachladen.

Nicht ganz so freizügig wirkt das „h" auf die Keyscan-Routine. Hier wird nur die Entprellschleife übergangen, die KSCAN normalerweise macht. Ein „s" bei diesen Optionen arbeitet beide Ausnahmen normal, also interpretativ, ab. Hier sollte daher immer ein „h" stehen.

Darunter sind die Einstellungen der drei verschiedenen Bildschirmfarben, wie oben beschrieben, zu finden. Die Werte für Vorder- und Hintergrundfarben entsprechen denen im BASIC, sind jedoch um 1 reduziert.

Die Optionen können durch Druck auf CTRL 9 auf die im Laufwerk 1 liegende EXPLORER-Diskette abgespeichert werden und sind beim nächsten Start wieder präsent. Achtung: Es erfolgt keine Sicherheitsabfrage – der Speichervorgang beginnt sofort.

DER REGISTER-BILDSCHIRM

FCTN 8 schaltet den Registerbildschirm an und aus. In der oberen Zone finden sich die Bit-für-Bit-Interpretationen von GPL-Statusbyte und VDP-Statusregister, wie sie in hexadezimaler Form im Hauptbildschirm stehen. Darunter sind die Werte aller VDP-Register ebenfalls wie im Hauptbildschirm zu sehen: links die Nummer des Registers, rechts daneben die duale Interpretation, gefolgt vom bekannten Hex-Wert. Es folgt die Funktion des Registers und der tatsächliche Wert im VDP-RAM, der sich aus der Funktion des VDP-Registers ergibt. Hier muß für weitere Information auf das Editor/Assembler-Handbuch verwiesen werden.

Eine nützliche Anwendung soll herausgestellt werden: Vor allem dann, wenn die Power-Up-Routinen des TI untersucht werden sollen, ist es sinnvoll und lehrreich, wenn die Bildschirmabschaltung aufgehoben wird, da alle Bildschirmdarstellungen mit dem Scan-Bit ('scn') auf Null (abgeschalteter Bildschirm) aufgebaut werden. Setzt man das Bit wieder auf „1", so kann der Bildschirmaufbau (CTRL 3 zum Umschalten) beobachtet werden. Hier kann der Breakpoint eingesetzt werden, um zu unterbrechen, wenn der Schirm abgeschaltet werden soll.

Ein Zugriff auf ein VDP-Register erfolgt über die Pseudo-Adresse > 8000. Setzt man „bp" des VDP-RAMs auf > 8180, wird eine Unterbrechung immer dann ausgelöst, wenn das Scan- und das Interrupt-Enable-Bit gelöscht werden sollen. Das läßt sich dann gleich korrigieren.

EXPLORER UND EXTENDED BASIC

Wird EXPLORER unter TI-Basic (mit E/A oder Minimen) benutzt, so entstehen keine Probleme hinsichtlich des Speicherbedarfs, da TI-BASIC nur das VDP-RAM und EXPLORER nur das CPU-RAM benutzt. Im Extended BASIC sieht das anders aus: Eigene XBASIC-Programme müssenmindestens 18400 Byte freien Program Space anzeigen (laufendes Programm mit FCTN 4 stoppen und dann SIZE eingeben), wenn EXPLORER noch in den Speicher passen soll.

Beim Einsatz von EXPLORER in XBASIC können daher verschiedene Probleme auftreten – immer vorausgesetzt, EXPLORER wird von dem XB-Programm mit CALL LOAD ("DSK1.XBEXP") geladen:

1. Das XBASIC-Programm ist mit einem Speicherbedarf von mehr als 12876 Byte zu lang. Das Programm muß also verkürzt werden.
2. Das XBASIC-Programm ist weniger als 12877 Byte lang, aber immer noch zu groß, um zusammen mit EXPLORER in die Speichererweiterung zu passen. Kein Problem – starten Sie das Programm XBMEMOFF von der Explorer-Diskette (vorher CALL FILES (1) eingeben), und ab sofort lädt XBASIC nur noch ins VDP-RAM.
3. Das XBASIC-Programm ist kurz genug, aber Ihr eventuell geladenes eigenes Maschinenprogramm füllt das gesamte Low-Memory, und Sie erhalten die Fehlermeldung „Memory Full", wenn EXPLORER geladen werden soll. Auch kein Problem – mergen Sie das File XBMEMFULL an Ihr XBASIC-Programm, und diese Fehlermeldung kommt nicht mehr.
4. Sie wollen, daß EXPLORER über Autostart lädt. Auch gut – nennen Sie das File XBLOAD,um in LOAD.

PROBLEME MIT DEM CPS - 99

EXPLORER will von einem Atronics-CPS-99 nichts wissen. Warum das so ist, soll ein kurzer Blick in den Loader klären.

Es hängt direkt mit dem Kopierschutz zusammen, für den eine Controller-Erkennung notwendig ist. Hier macht es sich der Loader zu EXPLORER sehr leicht. Er schaltet den Controller an, schaltet die zweite ROM-Bank an und prüft anhand des Byte <AA, das in der zweiten Bank an der Adresse <4000 nicht vorhanden sein sollte, ob wirklich eine zweite Bank existiert oder nicht.

Beim TI-Controller gibt es keine alternative ROM-Bank, beim CPS-99 aber sehr wohl, da hier die DSR zur Schnittstelle steht. Durch diesen Kunstgriff sparte man ein ROM ein und kommt, wie jeder weiß, der sein CPS schon mal zerlegt hat, mit einem 27128 aus. Leider gaukelt jedoch das CPS diesem Loader vor, er habe einen TI-Disk-Controller vor sich. Das Ergebnis ist, daß das Hauptprogramm nicht geladen werden kann.

Baut man diese Prüfung aus dem Loader aus, so kann man EXPLORER auch mit einem CPS-99 benützen. Hier die „Kochrezepte" für die beiden Loader (E/A und XB):

Am einfachsten geht es mit dem XB-Loader, da er mit dem Editor zum E/A-Modul geladen werden kann. In Zeile 5 wird die Sequenz

BAA00B1315B04E0BC
004BFF93B1F087F27A
in die Folge
BAA00B1000B04E0BC
004BFF93B1F088F27A
umgeändert.

Der Loader wird jetzt in keinem Fall mehr einen TI-Controller erkennen. Doch das ist kein Problem, da der originale Loader kopiert und unverändert aufgehoben werden kann.

Beim E/A-Loder ist die Sache etwas komplexer. Hier kann die Änderung mit einem Sektoreditor vorgenommen werden. Im ersten Sektor des Files EXP, normalerweise Nummer 34, werden die Byte

153 und 154 von > 1315 in > 1000 geändert. Anstatt des Prüfsummenbyte > 7 wird ein > 8 (gnore Checksum) oder ein > F (End Of Record) als Tag eingebaut. Wie man sieht, in Compressed Code. Auch von diesem Loader sollte vorher eine Kopie angefertigt werden.

EXPLORER UND DER MYARC 9640 GENEVE

EXPLORER startet nicht auf dem Geneve. Verantwortlich ist das recht eigensinnige Hardware-Konzept dieses Rechners, was die Einbindung externer Geräte in die Software angeht. Doch diese Eigenbrödlereien ist man von Myarc gewohnt. Da Myarc das gesamte DSR-Konzept zu Ehren des Geneve geopfert hat, kann von einem Programm aus dem Jahr 1985 nicht erwartet werden, zu wissen, daß man an die Adresse > 8002 ein > BA schreiben muß, um an den Disk-Controller zu kommen. Und woher soll so ein Programm wissen, daß das noch nicht alles war?

Geneve-Besitzer sind daher auf die schon länger kursierende geknackte Raubkopie von EXPLORER angewiesen, die daran erkennbar ist, daß sie die Hintergrundfarbe (back plane) des Hintergrund-Bildschirmes nicht korrekt setzt. Nach dem Laden mit E/A oder Minimem und sofortigem Druck auf CTRL 3 ist dies leicht zu erkennen. Auch lädt dieses Programm wesentlich langsamer als das Original.

RESUMEE

EXPLORER als Debugger zu bezeichnen, ist beinahe ein Sakrileg. Er ist noch viel mehr: Er ist das Werkzeug für all die Assembler-Schaffenden, die entweder ganz genau wissen wollen, was in ihrem Objekt der Begierde vorgeht, oder die den Anfang gerade hinter sich gelassen haben

BEIM KAUF AUF DAS .....

und praktisch erfahren möchten, was die abstrakten Bit und Byte im Handbuch zum Editor/Assembler-Modul bedeuten.

Wer das Programm kaufen will, der muß aufpassen, daß er nicht aus Versehen The Explorer, den Disk Editor von Tarik Isani, bestellt. Beide Programme kamen etwa zeitgleich auf den Markt, was das Namensproblem erklärt.

Die Anwendung von EXPLORER setzt mehr als nur Anfängerwissen um Struktur und Programmierung des TI voraus. Ein Lernwerkzeug ist er nicht.

Perfekt ist er auch nicht. Doch das liegt eher am TI als an EXPLORER. Es gibt eben keinen Speicher en masse, und die gelegentlich recht sinnvolle Hardcopy des Statusschirmes hätte den verfügbaren Platz für die zu simulierenden Programme reduziert. Immerhin stehen mit Minimem unter anderem das gesamte Low-Memory, das Mid-Memory im MMM und High-Mem von > A000 bis > B7D7 zur Verfügung.

..... RICHTIGE PRODUKT ACHTEN

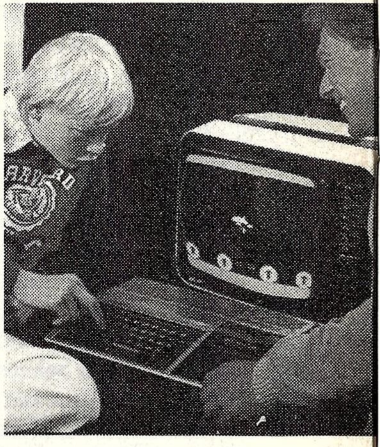
Negativ fällt der zu langsame Screen-Editor im Speicherfenster auf, auch die Cursorbewegnungen sind langsam. Aber man gewöhnt sich auch daran, schließlich handelt es sich um ein außergewöhnliches Programm.

Bezugsquellen sind die gleichen wie bei Diags, der Preis liegt derzeit bei etwa 20 US-Dollar.

Christopher Winter,
system-99 User-Group.

MODULEXPANDER FUNKTIONIERT NICHT

Ich habe den Modulexpander aus TI-REVUE 6/87 bzw. HCA 8/87 nachgebaut und er funktioniert nicht. Ich habe alles hundert mal kontrolliert, alles in Ordnung. Wenn ich GS vom IC 74LS139 abnehme und an IC 74LS139 pin 4 oder 5 lege, funktioniert Platz 1 einwandfrei; bei Anschluß an Pin 6 funktioniert Platz 2 einwandfrei. Mein Gedanke, Fehler im oder am 74LS139. Also IC getauscht und war nicht defekt. Nun simuliert, das heißt GS auf 0 Volt und A12 und A13 so auf Low und High gebracht, daß ich an den Modulstecker Pin 21 jeweils das Low-Signal für GS erhielt. Ich habe sogar die Leitungen A12 und A13 vertauscht, aber es werden die Module nicht angenommen



Probleme mit dem TI? Sie fragen, wir antworten

oder ich habe ein schwarzes Bild mit Ton. Bei der Vertauschung von A12 mit A13

```
(**********************************
* SETPLT(Nr,R,G,B:Integer)        *
* Setzt in den Screens 4..8       *
* die Palette Nr. auf die in      *
* den Parametern Rot,Gruen,Blau   *
* angegebenen Werte.              *
* Verwendung wie BASIC-Anweisung*
* COLOR = ( )                     *
**********************************)

procedure SetPlt(Nr,R,G,B:Integer);

VAR
    Work           : Byte;

BEGIN
Work:= R SHL 4;
Work:= Work OR B;
inline
(
 $3A/NR/            (* LD   A,(Nr)          *)
 $57/               (* LD   D,A             *)
 $3A/G/             (* LD   A,(G)           *)
 $5F/               (* LD   E,A             *)
 $3A/WORK/          (* LD   A,(Work)        *)
 $FD/$2A/$F7/$FA/(* LD   IY,(EXBRSA-1) *)
 $DD/$21/$4D/$01/(* LD   IX,SETPLT       *)
 $CD/$1C/$00        (* CALL CALSLT       *)
);
END;
```

war die Platine wieder im Rechner.
Wenn ich von Hand das GS-Signal auf Pin 4 oder 5 lege und dann mit Pin 6 tausche, kann ich zwischen den Modulen umschalten und es funktioniert einwandfrei, bei Anschluß laut Plan tut sich nichts oder aber es kommt schwarzes Bild mit Ton. Ich weiß auch, daß etwa die gleiche GS-Umschaltung im Gram-Modul selbst vorgenommen wurde, welches auch einwandfrei funktioniert. Kann es sein, daß ich A12 und A13 nicht zu den Modulsteckplätzen durchschleifen darf, was ich aber verstehen würde. Anzumerken wäre noch, daß ich A12 und A13 mittels zweier Kabel an das IC anschlossen habe, welche über den Leiterbahnen von Masse, CRUCLK, CRUIN und CRUOUT/A15 liegen.

Klaus Paltian
5650 Solingen 1

Prüfen Sie den Modulexpander aus Heft 6/87 zuerst mit reinen Grom-Modulen, die kein ROM oder EPROM im Bereich HEX 6000 aufweisen. Die Umschaltung mittels 7400 ist dabei inaktiv.
Wenn dies funktioniert, was wahrscheinlich ist, so überprüfen Sie die Verdrahtung des 7400 und 7432, eine Vertauschung von Anschlüssen würde genau zu Ihrer Fehlerbeschreibung passen.

BREMEN RIEF ZUM TI-TREFFEN, EINE STELLUNGNAHME

„Bremen rief zum TI-Treffen", TI-REVUE 10/87:
Die angesprochene IBM-Erweiterung für den TI-99/4A hat nicht zum Ziel, den uns ans Herz gewachsenen TI-99-Computer in irgendeiner Weise zu degradieren.
Auch gehört die abgebildete PC-Tastatur, die offensichtlich Teil des von einer anderen Firma vorgestellten Geneve-Zweitsysteme ist, nicht zum vorgestellten TRITON TURBO-XT-System, was aus den Abbildungen und der Unterschrift für den Leser des Artikels nicht klar genug hervorgeht.

Ziel der vorgestellten Erweiterung TRITON TURBO-XT ist vielmehr, jedem Benutzer des TI-99 unter Beibehaltung seines bisherigen TI-Systems, gleich welche Konfiguration auch immer vorliegt, die Möglichkeit der vollen IBM-Kompatibilität zu eröffnen, ohne einen kompletten zweiten Computerplatz einrichten zu müssen, für welchen meistens auch der Platz nicht vorhanden ist.
Das Einschleifen einer Bridge-Box in den TI-Bus und Zustellen einer zweiten Peripherie-Box TURBO-XT machen dies möglich.
Monitor und TI-Konsole mit allen vorhandenen Erweiterungen, wie Peribox, Diskettenstationen und Drucker können beibehalten werden.
Hätte sich der Schreiber des Artikels als freier Mitarbeiter Ihrer Zeitschrift zu erkennen gegeben, hätte er die Möglichkeit zu einer sauberen Recherche von uns bekommen. Dabei hätte er erfahren können, daß beide Systeme gleichzeitig arbeiten können (parallel processing). Um dies zu zeigen, wurden von uns zwei Monitore eingesetzt. Es ist wenig sinnvoll, zwei Bilder auf einen Monitor gleichzeitig darzustellen.
Normalerweise ist wohl nur ein Monitor vorhanden, der selbstverständlich beim Abarbeiten auf dem 8-MHz-TURBO-XT-System auch das entsprechende 80-Zeichen-Bild zeigen muß.
Es ist der verbliebenen TI-Gemeinde absolut nicht damit gedient, wenn durch Vermittlung persönlich gefärbter Halbwahrheiten eine Zensur der Leser vorgenommen wird, die bei ausführlicher Information selbst in die Lage gekommen wären, die besprochenen Produkte für ihren eigenen Bedarf zu beurteilen.
Nachdem der Hersteller TRITON absolut kein Interesse zeigte, den europäischen Markt mit seinem Produkt zu beliefern – es wurden sogar Schecks retourniert –, habe ich auf Drängen einiger TI-Freaks mit erheblichen Vorleistungen und Kosten erreicht, daß ich das TURBO-XT-System in Europa als einziger vorstellen und liefern kann. Sollte sich bei der nächsten Neuerung aus USA ein ähnlicher Trend des ungeprüften Zerreißens abzeichnen, können Sie sich meine Reaktion vorstellen. Sie werden in Zukunft keinen zweiten Lieferanten finden, der aus Idealismus für den TI-99 bereit ist, Investitionen in der erforderlichen Höhe für ein paar Dutzend Interessierte TI-Fans vorzunehmen.

Detlev Ernst Radiotechnik
5707 Pfinztal-1-Berghausen

Da die TI-Revue und auch ich als Autor des Bremen-Berichtes jeder sachlichen Kritik gegenüber offen sind, hätten wir diesen Brief auch ohne den Hinweis auf das Pressegesetz veröffentlicht. Nichts liegt mir ferner als eine „Zensur der Leser". Ich würde mich sogar freuen, wenn sich bei den im Brief erwähnten ausführlichen Informationen zeigen würde, daß ich die Verbindung TI-99/4A - TRITON TURBO-XT unterschätzt habe. Ich hoffe, daß die TI-Revue in einer der nächsten Ausgaben eine ausführliche Beschreibung dieses Systems veröffentlichen kann. Auch Erfahrungsberichte von TI-Usern, die bereits mit arbeiten, wären für die Leser dieser Zeitschrift sicherlich von Interesse.

Jürgen Stelter

NEUES E/A-MODUL

In der TI-Revue 11 (HCA 2/88) haben wir über den neuen Editor/Assembler von Winfried Winkler berichtet. In der Zwischenzeit hat der Autor uns mitgeteilt, daß die endgültig letzte Version #6 fertig sei.

Eine weitere Verbesserung besteht darin, daß die Files EDIT1, ASSM1 und ASSM2 in Form von GROM-Files mit im Modul enthalten sind und somit nicht mehr von Diskette geladen werden müssen. Damit brauchen die TI-User, die nur ein Laufwerk besitzen, nicht jedesmal die E/A-System-Diskette wieder einzulegen, wenn sie aus dem Editor in den Assembler wechseln wollen oder umgekehrt. Selbst wenn sie mehrere Laufwerke benutzen, geht das Überwechseln von einem in den anderen E/A-Modus um ein Vielfaches schneller.

Weiterhin wurde der LOADER um die Möglichkeit erweitert, alle DEFinierten Startnamen auf dem Bildschirm anzuzeigen (unresolved REFerences sind markiert). Auch können die Bildschirmfarben vom User festgelegt werden. Das Modul ist wesentlich komfortabler und, was besonders wichtig ist, auch um einiges schneller als das Vorbild von Texas Instruments. Allerdings wird diese „lange" Version nur noch unter Angabe der Ladesoftware vom Autor abgegeben, da alle verfügbaren Versionen (siehe TI-Revue 11, Seiten 35 bis 36) zusammen mehr als drei Disketten füllen würden.

Die Adresse für Interessenten an dieser Fairware:

Winfried Winkler
Rathenower Straße 69
1000 Berlin 21
Tel. 030/394 47 49

```
100 !        **************
110 !
120 !      *** SCHACH86 ***
130 !
140 !     *** VERSION II ***
150 !
160 !   *** COPYRIGHT BY ***
170 !
180 !   *** JOERG LESSING ***
190 !
200 ! **********************
21Ø CALL SCREEN(2):: CALL HC
HAR(1,1,1Ø1,768):: FOR I=32
TO 63 :: READ I$ :: CALL CHA
R(I,I$):: NEXT I :: FOR I=64
 TO 91 :: READ I$ :: CALL CH
AR(I,"ØØ"&I$):: NEXT I :: FO
R I=92 TO 143 :: READ I$ ::
CALL CHAR(I,I$):: NEXT I
22Ø DATA FFFFFEFDFBF7FBFD,FF
FF7FBFDFEFDFBF,ØØØØØ1Ø3Ø7ØFØ
7Ø3,ØØØØ8ØCØEØFØEØC,FFFFFEFC
F8FØF8FC,FFFF7F3F1FØF1F3F,ØØ
ØØØ1Ø2Ø4Ø8Ø4Ø2,ØØØØ8Ø4Ø2Ø1Ø2
Ø4
23Ø DATA FFFEFEFDF9FEFEFD,FF
7F7FBF9F7F7FBF,ØØØ1Ø1Ø3Ø7Ø1Ø
1Ø3,ØØ8Ø8ØCØEØ8Ø8ØC,FFFEFEFC
F8FEFEFC,FF7F7F3F1F7F7F3F,ØØ
Ø1Ø1Ø2Ø6Ø1Ø1Ø2,ØØ8Ø8Ø4Ø6Ø8Ø8
Ø4
24Ø DATA ØØ7C4C5464447C,ØØ7Ø
1Ø1Ø1Ø1Ø7C,ØØ7CØ4Ø47C4Ø7C,ØØ
7CØ47CØ4Ø47C,ØØ4444447CØ4Ø4,
ØØ7C4Ø4Ø7CØ47C,ØØ7C4Ø7C44447
C,ØØ7CØ4Ø4Ø4Ø4Ø4,ØØ7C447C444
47C
25Ø DATA ØØ7C44447CØ47C,ØØ2Ø
6ØFE6Ø2,ØØØ8ØCFEØCØ8,Ø1Ø3Ø1Ø
ØØ7Ø3Ø1ØF,Ø3Ø1Ø1Ø3ØF1F1F,8ØC
Ø8ØØØEØCØ8ØF,CØ8Ø8ØCØFØF8F8
26Ø DATA ,7C44447C4444,78447
8444478,7C4Ø4Ø4Ø4Ø7C,7844444
44478,7C4Ø784Ø4Ø7C,7C4Ø784Ø4
Ø4,7C4Ø5C44447C,44447C444444
,381Ø1Ø1Ø1Ø38,7CØ4Ø4Ø44438
27Ø DATA 485Ø6Ø5Ø4844,4Ø4Ø4Ø
4Ø4Ø7C,446C54444444,4464544C
4444,7C444444447C,7844784Ø4Ø
4,7C4444544C7C,7844785Ø4844,
7C4Ø7CØ4Ø47C,7C1Ø1Ø1Ø1Ø1Ø1
28Ø DATA 44444444447C,444428
281Ø1,44445454547C,44281Ø284
444,4444281Ø1Ø1,7CØ81Ø2Ø4Ø7C
,ØØ1ØØØØØ1
29Ø DATA Ø1Ø3Ø1ØØØ7Ø3Ø1ØF,Ø3
Ø1Ø1Ø3ØF1F1F,8ØCØ8ØØØEØCØ8ØF
,CØ8Ø8ØCØFØF8F8,FFFFCØCØCØCØ
CØC,CØCØCØCØCØCØFFFF,FFFFØ3Ø
3Ø3Ø3Ø3Ø3,Ø3Ø3Ø3Ø3Ø3Ø3FFFF
3ØØ DATA FFFFFFFFFFFFFFFF,,,
,ØØØ1ØØØ1Ø3Ø3Ø1ØF,Ø3Ø1Ø1Ø3ØF
1F1F,ØØ8ØØØ8ØCØCØ8ØF,CØ8Ø8ØC
ØFØF8F8,ØØØ1ØØØ1Ø3Ø3Ø1ØF

31Ø DATA Ø3Ø1Ø1Ø3ØF1F1F,ØØ8Ø
ØØ8ØCØCØ8ØF,CØ8Ø8ØCØFØF8F8,F
FC4D5D1EFF7FBFB,FF23AB8BF7EF
DFDF,ØØ3B3B3F1FØFØ7Ø7,ØØDCDC
FCF8FØEØE
32Ø DATA FFC4C4CØEØFØF8F8,FF
4343Ø3Ø7ØF1F1F,ØØ3B2A2E1ØØ8Ø
4Ø4,ØØDC54748Ø81Ø2Ø2,FCF3EFF6
F7FBFBFD,3FCF1FFF7FBFDFBF,Ø3
ØF1FØFØFØ7Ø7Ø3,CØFØEØØØ8ØCØE
ØC
33Ø DATA FCFØEØFØFØF8F8FC,3F
ØF1FFF7F3F1F3F,Ø3ØC1ØØ9Ø8Ø4Ø
4Ø2,CØ3ØEØØØ8Ø4Ø2Ø4
34Ø DATA FBFBFBFBF7EFEØFF,DF
DFDFDFEFF7Ø7FF,Ø7Ø7Ø7Ø7ØF1F1
F,EØEØEØEØFØF8F8,F8F8F8F8FØE
ØEØFF,1F1F1F1FØFØ7Ø7FF,Ø4Ø4Ø
4Ø4Ø81Ø1F,2Ø2Ø2Ø2Ø1ØØ8F8
35Ø DATA FEFEFDFBF7EFEØFF,7F
7FBFDFEFF7Ø7FF,Ø1Ø1Ø3Ø7ØF1F1
F,8Ø8ØCØEØFØF8F8,FEFEFCF8FØE
ØEØFF,7F7F3F1FØFØ7Ø7FF,Ø1Ø1Ø
2Ø4Ø81Ø1F,8Ø8Ø4Ø2Ø1ØØ8F8
36Ø DATA @@@@ee@@@@eeGESAMTZ
UGZAHL[ØØØe,@@@@de@@@@de@SCH
ACH86@BY@YSC@d,@@@@de@@@@dee
dddddddddddddddd
362 ! In den Zeilen 37Ø,38Ø,
4ØØ und 41Ø die Zeichen, die
 zwischen "^" stehen, mit CT
RL eingeben. Die "^" koennen
 entfallen.
364 ! In Zeile 41Ø in das "V
" zwischen den beiden "^" mi
t FCTN einzugeben. Die "^" k
oennen entfallen.
366 ! In dem letzten String
in Zeile 37Ø koennen die Anf
uehrungszeichen weggelassen
werden, wenn das Komma mit C
TRL eingegeben worden ist.
368 ! Alle uebrigen Leerzeic
hen koennen mit der Spacetas
te eingegeben werden.
37Ø DATA @@@@de@@@@deeeeeeee
eeeeeeeeeee,@@@@de@@@@dersxy
*+^PPPP^()z{pqee,"@@@@de@@@@
de^BCHIJKPPPPHIJK,A^8e"
38Ø DATA @@@@de@@@@de !"# !"
# !"# !"#@d,eddddeedddde^HIJ
KHIJKHIJKHIJK^7d,eeeeeeeeeee
e                 @d,;@@@@@@@
```

```
@:ee                6d
390 DATA @@@@@@@@@@@de
         @d,ZUG@;@@@@:de
            5d,@@@@@@@@@@@de
               @d,ZEIT@ØØ[ØØ
de                  4d
400 DATA eddddddddde
         @d,eeeeeeeeeeee
            3d,¦@@@@@@@@@ee&
'$%&'$%&'$%&'$%@d,@@@@@@@@@@
de^NOLMNOLMNOLMNOLM^2d

410 DATA ZUG@;@@@@:detu~^V^^
P^-^PPPP^./¦}vw@d,@@@@@@@@@@
de^DENOLMPPPPNOLMFG^1d,ZEIT@
ØØ[ØØdeeA@B@C@D@E@F@G@H@d,ed
ddddddddeeeddddddddddddddd
420 CALL MAGNIFY(3):: FOR I=
2 TO 23 :: READ I$ :: CALL H
CHAR(I,2,ASC(SEG$(I$,1,1))):
: DISPLAY AT(I,1):SEG$(I$,2,
28):: CALL HCHAR(I,31,ASC(SE
G$(I$,3Ø,1)))
43Ø NEXT I :: CALL INIT :: O
N BREAK NEXT :: CALL LOAD(-3
18Ø6,16):: CALL HCHAR(2Ø,18,
44):: CALL HCHAR(2Ø,2Ø,3Ø,2)
:: CALL HCHAR(21,2Ø,3Ø,2)::
J$="@@@@@@@@"
435 ! Die beiden Leerzeichen
 in den Anfuehrungszeichen i
n den Zeilen 44Ø und 45Ø sin
d durch "CTRL P" zu ersetzen
.
44Ø CALL HCHAR(6,22,3Ø,2)::
CALL HCHAR(7,22,3Ø,2):: I$=C
HR$(3Ø)&CHR$(3Ø)&"" :: FOR
 I=1Ø TO 14 STEP 4 :: WB$,SB
$="1111111"
45Ø DISPLAY AT(I,12):I$&I$&I
$&I$;:: DISPLAY AT(I+1,12):I
$&I$&I$&I$;:: DISPLAY AT(I+2
,12):""&I$&I$&I$&SEG$(I$,1
,2);:: DISPLAY AT(I+3,12):"
"&I$&I$&I$&SEG$(I$,1,2);
46Ø NEXT I :: DIM K$(8,8),Z$
(1,7):: FOR I=1 TO 8 :: FOR
II=1 TO 8 :: READ K$(I,II)::
 NEXT II :: NEXT I :: FOR I=
Ø TO 1 :: FOR II=1 TO 7 :: Z
$(I,II)="@@@@" :: NEXT II ::
 NEXT I :: J$(1)="12345678AB
CDEFGHKLNORSV"
47Ø DATA TWS,PWW,LWS,DWW,KWS
,LWW,PWS,TWW,BWW,BWS,BWW,BWS
,BWW,BWS,BWW,BWS,FSS,FWW,FSS
,FWW,FSS,FWW,FSS,FWW
48Ø DATA FWW,FSS,FWW,FSS,FWW
,FSS,FWW,FSS,FSS,FWW,FSS,FWW
,FSS,FWW,FSS,FWW,FWW,FSS,FWW
,FSS,FWW,FSS,FWW,FSS
49Ø DATA BSS,BSW,BSS,BSW,BSS
,BSW,BSS,BSW,TSW,PSS,LSW,DSS
,KSW,LSS,PSW,TSS
5ØØ F$(Ø)="Ø1Ø3Ø1ØØØ7Ø3Ø1ØFØ
3Ø1Ø1Ø3ØF1F1FØØ8ØCØ8ØØØEØCØ8
ØFØCØ8Ø8ØCØFØF8F8ØØ" :: F$(1
)="Ø1Ø2Ø1ØØØ7Ø2Ø1ØEØ2Ø1Ø1Ø2Ø
C1Ø1FØØ8Ø4Ø8ØØØEØ4Ø8Ø7Ø4Ø8Ø8
Ø4Ø3ØØ8F8ØØ"
51Ø F$(2)="ØØØ1ØØØ1Ø3Ø3Ø1ØFØ
3Ø1Ø1Ø3ØF1F1FØØØØ8ØØØ8ØCØCØ8
ØFØCØ8Ø8ØCØFØF8F8ØØ" :: F$(3
)="ØØØ1ØØØ1Ø2Ø2Ø1ØEØ2Ø1Ø1Ø2Ø
C1Ø1FØØØØ8ØØØ8Ø4Ø4Ø8Ø7Ø4Ø8Ø8
Ø4Ø3ØØ8F8ØØ"

52Ø CALL CHAR(1Ø8,F$(3),1Ø4,
F$(3)):: CALL SPRITE(#3,6Ø,2
,41,169,#4,1Ø8,16,41,153,#5,
92,16,153,169,#6,1Ø4,2,153,1
53)
53Ø FOR I=3 TO 8 :: CALL COL
OR(Ø,16,16,1,2,16,2,2,16,I,7
,12,9,13,4,11,2,16,12,2,16,1
3,2,16,14,2,16):: NEXT I ::
FOR I=Ø TO 1
54Ø ACCEPT AT(I*7+11,1)VALID
ATE(UALPHA,"@")SIZE(-8):N$(I
):: NEXT I :: DISPLAY AT(3,1
2):"ZUGOBERGRENZE[ØØØ" :: AC
CEPT AT(3,26)SIZE(-3)VALIDAT
E(DIGIT):ZO :: K,L,M,N=1 ::
Y=2Ø :: X=14 :: J$(Ø)="@ABCD
EHKLNORSV"
55Ø DISPLAY AT(3,12):"SPIELE
R@1@Ø;J@1;K" :: CALL KEY(Ø,L
1,L2):: IF L1<48 OR L1>49 TH
EN 55Ø ELSE JK(Ø)=L1-48 :: C
ALL WAIT(5Ø)
56Ø DISPLAY AT(3,12):"SPIELE
R@2@Ø;J@1;K" :: CALL KEY(Ø,L
1,L2):: IF L1<48 OR L1>49 TH
EN 56Ø ELSE CALL HCHAR(3,14,
64,17):: JK(1)=L1-48
57Ø CALL HCHAR(SP*7+13,7,64,
4):: CALL DELSPRITE(#2):: CA
LL U(SO,SP,SQ):: IF JK(SP)=1
 THEN 66Ø ELSE CALL SPRITE(#
1,96,5,Y*8-7,X*8-7)
58Ø CALL JOYST(SP+1,L1,L2)::
 CALL KEY(SP+1,L3,L4):: IF L
1=-4 THEN 59Ø ELSE IF L1=4 T
HEN 6ØØ ELSE IF L2=4 THEN 61
Ø ELSE IF L2=-4 THEN 62Ø ELS
E IF L3=18 THEN 64Ø ELSE IF
L3=11 AND G=Ø THEN 66Ø ELSE
63Ø
```

```
590 X=X+2*(X>14):: IF G=0 TH
EN L=L+1*(L>1):: GOTO 630 EL
SE N=N+1*(N>1):: GOTO 630
600 X=X-2*(X<28):: IF G=0 TH
EN L=L-1*(L<8):: GOTO 630 EL
SE N=N-1*(N<8):: GOTO 630
610 Y=Y+2*(Y>6):: IF G=0 THE
N K=K-1*(K<8):: GOTO 630 ELS
E M=M-1*(M<8):: GOTO 630
620 Y=Y-2*(Y<20):: IF G=0 TH
EN K=K+1*(K>1):: GOTO 630 EL
SE M=M+1*(M>1)
630 CALL LOCATE(#1,Y*8-7,X*8
-7):: GOTO 580
640 IF G=0 THEN CALL SPRITE(
#2,96,6,Y*8-7,X*8-7):: CALL
COLOR(#1,8):: CALL WAIT(99):
: CALL COLOR(#1,5):: G=1 ::
M=K :: N=L :: GOTO 580 ELSE
G=0
650 CALL LOCATE(#1,Y*8-7,X*8
-7):: E$(SP)=CHR$(L+64)&CHR$
(K+48)&CHR$(N+64)&CHR$(M+48)
:: K1=M :: L1=N :: GOTO 730
660 CALL DELSPRITE(#1,#2)::
ACCEPT AT(SP*7+13,5)VALIDATE
(J$(JK(SP)))SIZE(-4):E$(SP):
: IF E$(SP)="@@@@" OR E$(SP)
="" THEN 660 ELSE IF E$(SP)=
"ROCH" AND SP=0 THEN 680
670 IF E$(SP)="ROCH" THEN 70
0 ELSE IF E$(SP)="ENDE" THEN
 1060 ELSE IF E$(SP)="SAVE"
AND SP=0 THEN 1080 ELSE IF E
$(SP)="LOAD" AND SP=0 THEN 1
110 ELSE IF JK(SP)=1 THEN ON
 ERROR 660 :: GOTO 720
680 IF Q(SP)=0 AND K$(1,6)="
FWW" AND K$(1,7)="FSS" AND K
$(1,5)="KWS" AND K$(1,8)="TW
W" THEN K$(1,5)="FSS" :: Q(S
P)=1 :: GOTO 690 ELSE CALL F
1 :: GOTO 570
690 K1=K :: L1=L :: K$(1,6)=
"TWW" :: K$(1,7)="KWS" :: K$
(1,8)="FWW" :: CALL N(1,8,30
):: CALL D(1,6,118,134):: CA
LL S(1,7,16,92,5,F$(0),K$(,)
):: GOTO 870
700 IF Q(SP)=0 AND K$(8,6)="
FSS" AND K$(8,7)="FWW" AND K
$(8,5)="KSW" AND K$(8,8)="TS
S" THEN K$(8,5)="FWW" :: Q(S
P)=1 :: GOTO 710 ELSE CALL F
1 :: GOTO 570
710 K1=K :: L1=L :: K$(8,6)=
"TSS" :: K$(8,7)="KSW" :: K$
(8,8)="FSS" :: CALL N(8,8,14
4):: CALL D(8,6,112,128):: C
ALL S(8,7,2,60,3,F$(0),K$(,)
):: GOTO 870
720 K=VAL(SEG$(E$(SP),2,1)):
: L=ASC(SEG$(E$(SP),1,1))-64
 :: M=VAL(SEG$(E$(SP),4,1)):
: N=ASC(SEG$(E$(SP),3,1))-64
730 IF SEG$(K$(K,L),1,1)="F"
 THEN 830 ELSE IF SP=0 AND S
EG$(K$(K,L),2,1)="S" OR SP=1
 AND SEG$(K$(K,L),2,1)="W" T
HEN CALL F2 :: GOTO 840 ELSE
 IF SP=0 AND SEG$(K$(M,N),2,
1)="W" AND SEG$(K$(M,N),1,1)
<>"F" THEN 830
740 IF SP=1 AND SEG$(K$(M,N)
,2,1)="S" AND SEG$(K$(M,N),1
,1)<>"F" THEN 830 ELSE IF K=
M AND L=N THEN 830 ELSE CALL
 Y(K,L,M,N,K$(,),ZW):: IF ZW
=0 THEN CALL F1 :: GOTO 840
ELSE H3$=SEG$(K$(M,N),1,1)
750 IF K$(K,L)="BWW" OR K$(K
,L)="BWS" THEN IF ((K+1=M AN
D L=N AND H3$="F")+(K+2=M AN
D L=N AND K=2 AND H3$="F")-(
(K+1=M AND L+1=N AND H3$<>"F
")+(K+1=M AND L-1=N AND H3$<
>"F")))THEN 850 ELSE 830
760 IF K$(K,L)="BSS" OR K$(K
,L)="BSW" THEN IF ((K-1=M AN
D L=N AND H3$="F")+(K-2=M AN
D L=N AND K=7 AND H3$="F")-(
(K-1=M AND L+1=N AND H3$<>"F
")+(K-1=M AND L-1=N AND H3$<
>"F")))THEN 850 ELSE 830
770 IF K$(K,L)="TWS" OR K$(K
,L)="TWW" OR K$(K,L)="TSW" O
R K$(K,L)="TSS" THEN IF (L=N
 AND(M<K XOR M>K))-(K=M AND(
N<L XOR N>L))THEN 850 ELSE 8
30
780 IF K$(K,L)="PWW" OR K$(K
,L)="PWS" OR K$(K,L)="PSS" O
R K$(K,L)="PSW" THEN IF ((K+
2=M OR K-2=M)*(L-1=N OR L+1=
N))-((K+1=M OR K-1=M)*(L-2=N
 OR L+2=N))THEN 850 ELSE 830
790 IF K$(K,L)="KWS" OR K$(K
,L)="KWW" OR K$(K,L)="KSS" O
R K$(K,L)="KSW" THEN IF ((K+
1=M OR K-1=M)*(L=N OR(L-1=N
XOR L+1=N)))+((L+1=N OR L-1=
N)*(K=M OR(K-1=M XOR K+1=M))
)THEN 850 ELSE 830
800 IF K$(K,L)="DWS" OR K$(K
,L)="DWW" OR K$(K,L)="DSS" O
R K$(K,L)="DSW" THEN IF ((K+
```

```
1=M OR K-1=M)*(L=N OR(L-1=N
XOR L+1=N)))+((L+1=N OR L-1=
N)*(K=M OR(K-1=M XOR K+1=M))
)THEN 85Ø
81Ø IF K$(K,L)="DWS" OR K$(K
,L)="DWW" OR K$(K,L)="DSS" O
R K$(K,L)="DSW" THEN IF (L=N
 AND(M<K XOR M>K))-(K=M AND(
N<L XOR N>L))THEN 85Ø
82Ø IF (K$(K,L)="DWS" OR K$(
K,L)="DWW" OR K$(K,L)="DSS"
OR K$(K,L)="DSW")-(K$(K,L)="
LWS" OR K$(K,L)="LWW" OR K$(
K,L)="LSS" OR K$(K,L)="LSW")
THEN IF (ABS(L-N)=ABS(K-M))T
HEN 85Ø ELSE 83Ø
83Ø CALL F :: K=K1 :: L=L1 :
: CALL HCHAR(SP*7+13,7,64,4)
:: GOTO 57Ø
84Ø K=K1 :: L=L1 :: CALL HCH
AR(SP*7+13,7,64,4):: GOTO 57
Ø
85Ø DISPLAY AT(SP*7+13,5):E$
(SP);:: H1$=SEG$(K$(K,L),1,2
):: H2$=SEG$(K$(K,L),3,1)
86Ø K$(K,L)="F"&H2$&H2$ :: K
$(M,N)=H1$&SEG$(K$(M,N),3,1)
:: IF SEG$(K$(K,L),3,1)="S"
THEN CALL N(K,L,144)ELSE CAL
L N(K,L,3Ø)
87Ø CALL Z(M,N,F$(),K$(,))::
 FOR I=2 TO 7 :: Z$(SP,I-1)=
Z$(SP,I):: NEXT I :: Z$(SP,7
)=E$(SP):: CALL X(Z$(,),SP)

88Ø CALL K :: FOR I=1 TO 8 :
: M=I :: IF (K$(1,I)="BSS" O
R K$(1,I)="BSW")AND SEG$(SB$
,I,1)="1" THEN @=1 :: GOTO 9
ØØ ELSE IF (K$(8,I)="BWW" OR
 K$(8,I)="BWS")AND SEG$(WB$,
I,1)="1" THEN @=8 :: GOTO 9Ø
Ø
89Ø NEXT I :: GOTO 1Ø4Ø
9ØØ DISPLAY AT(3,12):"NEUE@S
PIELFIGUR@T" :: ACCEPT AT(3,
28)SIZE(-1)VALIDATE("TLPD"):
@$ :: IF @$="" THEN 9ØØ ELSE
 CALL HCHAR(3,14,64,17):: H5
$=SEG$(K$(@,M),2,2):: H6$=@$
&H5$
91Ø WI=Ø :: FOR I=1 TO 8 ::
FOR II=1 TO 8 :: IF SEG$(K$(
I,II),1,2)=SEG$(H6$,1,2)THEN
 WI=WI+1 :: IF WI=1 AND(SEG$
(H6$,1,2)="DW" OR SEG$(H6$,1
,2)="DS")THEN 93Ø
92Ø NEXT II :: NEXT I :: IF
WI=1 OR WI=Ø THEN K$(@,M)=H6
$ :: GOTO 96Ø
93Ø DISPLAY AT(3,12):"@@HABE
@NOCH@"&STR$(WI)&"@"&@$&"@@"
 :: CALL WAIT(99):: CALL HCH
AR(3,14,64,17):: IF @=1 THEN
 94Ø ELSE 95Ø
94Ø SB$=SEG$(SB$,1,M-1)&"2"&
SEG$(SB$,M+1,M-I):: GOTO 1Ø4
Ø
95Ø WB$=SEG$(WB$,1,M-1)&"2"&
SEG$(WB$,M+1,8-M):: GOTO 1Ø4
Ø
96Ø H4$=SEG$(K$(@,M),3,1)::
IF H4$="W" AND @$="T" AND @=
8 THEN CALL D(@,M,118,134)::
 GOTO 1Ø4Ø ELSE IF H4$="S" A
ND @$="T" AND @=8 THEN CALL
D(@,M,116,132):: GOTO 1Ø4Ø
97Ø IF H4$="W" AND @$="T" AN
D @=1 THEN CALL D(@,M,114,13
Ø):: GOTO 1Ø4Ø ELSE IF H4$="
S" AND @$="T" AND @=1 THEN C
ALL D(@,M,112,128):: GOTO 1Ø
4Ø
98Ø IF H4$="W" AND @$="L" AN
D @=8 THEN CALL D(@,M,46,142
):: GOTO 1Ø4Ø ELSE IF H4$="S
" AND @$="L" AND @=8 THEN CA
LL D(@,M,44,14Ø):: GOTO 1Ø4Ø
99Ø IF H4$="W" AND @$="L" AN
D @=1 THEN CALL D(@,M,42,138
):: GOTO 1Ø4Ø ELSE IF H4$="S
" AND @$="L" AND @=1 THEN CA
LL D(@,M,4Ø,136):: GOTO 1Ø4Ø
1ØØØ IF H4$="W" AND @$="P" A
ND @=8 THEN CALL D(@,M,126,1
42):: GOTO 1Ø4Ø ELSE IF H4$=
"S" AND @$="P" AND @=8 THEN
CALL D(@,M,124,14Ø):: GOTO 1
Ø4Ø

1Ø1Ø IF H4$="W" AND @$="P" A
ND @=1 THEN CALL D(@,M,122,1
38):: GOTO 1Ø4Ø ELSE IF H4$=
"S" AND @$="P" AND @=1 THEN
CALL D(@,M,12Ø,136):: GOTO 1
Ø4Ø
1Ø2Ø IF H4$="W" AND @$="D" A
ND @=8 THEN CALL S(@,M,2,1Ø4
,6,F$(3),K$(,)):: GOTO 1Ø4Ø
ELSE IF H4$="S" AND @$="D" A
ND @=8 THEN CALL S(@,M,16,1Ø
4,6,F$(2),K$(,)):: GOTO 1Ø4Ø
1Ø3Ø IF H4$="W" AND @$="D" A
ND @=1 THEN CALL S(@,M,2,1Ø8
,4,F$(2),K$(,)):: GOTO 1Ø4Ø
ELSE IF H4$="S" AND @$="D" A
ND @=1 THEN CALL S(@,M,16,1Ø
8,4,F$(3),K$(,))
```

```
1Ø4Ø CALL HCHAR(SP*7+11,11,6
4):: CALL HCHAR(SP*7+13,7,64
,4):: DISPLAY AT(SP*7+13,5):
"@@@@";:: IF SP=Ø THEN SP=1
ELSE SP=Ø
1Ø5Ø CALL HCHAR(SP*7+11,11,5
8):: K=K1 :: L=L1 :: IF SP=1
 THEN 57Ø ELSE ZU=ZU+1 :: CA
LL J(ZU):: IF ZU>=ZO AND ZO>
Ø THEN 1Ø6Ø ELSE IF ZU<ZO OR
 ZO=Ø THEN 57Ø
1Ø6Ø DISPLAY AT(3,12):"@ENDE
@DER@PARTIE@" :: CALL WAIT(3
5Ø):: DISPLAY AT(3,12):"@@DR
UECKE@F5@F9@@"
1Ø7Ø CALL KEY(Ø,T1,T2):: IF
T1=15 THEN CALL DELSPRITE(AL
L):: CALL CLEAR :: END ELSE
IF T1=14 THEN RUN ELSE 1Ø7Ø
1Ø8Ø DISPLAY AT(3,12):"NAME[
"&]$&"@@@@" :: ACCEPT AT(3,1
7)SIZE(-8)VALIDATE(UALPHA,DI
GIT):]$ :: CALL HCHAR(3,14,6
4,17):: ON ERROR 1Ø8Ø :: OPE
N #1:"DSK1."&]$&"_S",INTERNA
L,OUTPUT,VARIABLE 254 :: FOR
 I=1 TO 8
1Ø9Ø PRINT #1:K$(I,1),K$(I,2
),K$(I,3),K$(I,4),K$(I,5),K$
(I,6),K$(I,7),K$(I,8):: NEXT
 I :: FOR I=Ø TO 1 :: FOR II
=1 TO 7 :: PRINT #1:Z$(I,II)
,:: NEXT II :: NEXT I
11ØØ PRINT #1:ZO,ZU,SO,SQ,JK
(Ø),JK(1):: CLOSE #1 :: GOTO
 57Ø
111Ø DISPLAY AT(3,12):"NAME[
"&]$&"@@@@" :: ACCEPT AT(3,1
7)SIZE(-8)VALIDATE(UALPHA,DI
GIT):]$ :: CALL HCHAR(3,14,6
4,17):: ON ERROR 111Ø :: OPE
N #1:"DSK1."&]$&"_S",INTERNA
L,INPUT ,VARIABLE 254 :: FOR
 I=1 TO 8

112Ø INPUT #1:K$(I,1),K$(I,2
),K$(I,3),K$(I,4),K$(I,5),K$
(I,6),K$(I,7),K$(I,8):: NEXT
 I :: FOR I=Ø TO 1 :: FOR II
=1 TO 7 :: INPUT #1:Z$(I,II)
,:: NEXT II :: NEXT I
113Ø INPUT #1:ZO,ZU,SO,SQ,JK
(Ø),JK(1):: CLOSE #1 :: DISP
LAY AT(3,12):"ZUGOBERGRENZE[
"&RPT$("Ø",3-LEN(STR$(ZO)))&
STR$(ZO):: CALL WAIT(25Ø)::
CALL HCHAR(3,14,64,17):: CAL
L DELSPRITE(ALL)
1135 ! Die Leerzeichen in de
n Anfuehrungszeichen in der
Zeile 114Ø muessen durch "CT
RL P" ersetzt werden.
114Ø I1$=CHR$(3Ø)&CHR$(3Ø)&"
" :: FOR I=6 TO 18 STEP 4
:: FOR II=Ø TO 1 :: DISPLAY
AT(I+II,12):I1$&I1$&I1$&I1$;
:: DISPLAY AT(I+II+2,12):"
"&I1$&I1$&I1$&SEG$(I1$,1,2);
:: NEXT II :: NEXT I :: SP=Ø
115Ø FOR I=8 TO 1 STEP -1 ::
 FOR II=1 TO 8 :: CALL Z(I,I
I,F$(),K$(,)):: NEXT II :: N
EXT I :: FOR I=Ø TO 1 :: CAL
L X(Z$(,),I):: NEXT I :: CAL
L J(ZU):: Y=2Ø :: X=14 :: K,
L,M,N=1 :: GOTO 57Ø
116Ø SUB F :: CALL SOUND(2ØØ
,22Ø,Ø):: DISPLAY AT(3,12):"
@FALSCHE@EINGABE@" :: CALL W
AIT(25Ø):: CALL HCHAR(3,14,6
4,17):: SUBEND
117Ø SUB N(A,B,C):: CALL HCH
AR(22-A*2,12+B*2,C,2):: CALL
 HCHAR(23-A*2,12+B*2,C,2)::
SUBEND
118Ø SUB D(A,B,C,D):: DISPLA
Y AT(22-A*2,1Ø+B*2):CHR$(C)&
CHR$(C+1);:: DISPLAY AT(23-A
*2,1Ø+B*2):CHR$(D)&CHR$(D+1)
;:: FOR I=3 TO 6 :: CALL POS
ITION(#I,U1,U2)
119Ø IF INT((U1+7)/8)=22-A*2
 AND INT((U2+7)/8)=12+B*2 TH
EN CALL DELSPRITE(#I):: SUBE
XIT
12ØØ NEXT I :: SUBEND
121Ø SUB S(A,B,C,D,E,A$,K$(,
)):: CALL SPRITE(#E,D,C,(22-
A*2)*8-7,(12+B*2)*8-7):: CAL
L CHAR(D,A$):: FOR I=3 TO 6
:: IF I=E THEN 123Ø ELSE CAL
L COINC(#E,#I,5,U1)
122Ø IF U1=-1 THEN CALL DELS
PRITE(#I):: SUBEXIT
123Ø NEXT I :: CALL GCHAR(22
-A*2,12+B*2,U1):: IF U1=144
OR U1=3Ø THEN SUBEXIT ELSE I
F SEG$(K$(A,B),3,1)="S" THEN
 CALL N(A,B,144):: SUBEXIT E
LSE CALL N(A,B,3Ø)

124Ø SUBEND
125Ø SUB K :: CALL POSITION(
#3,U1,U2,#5,U3,U4):: IF U1>Ø
 AND U2>Ø AND U3>Ø AND U4>Ø
THEN SUBEXIT ELSE IF U3=Ø AN
D U4=Ø THEN DISPLAY AT(3,12)
:"SCHWARZ@IST@CHAMP" :: CALL
```

```
WAIT(999):: GOTO 1270
1260 DISPLAY AT(3,12):"@WEIS
S@IST@CHAMP@" :: CALL WAIT(9
99)
1270 DISPLAY AT(3,12):"@@DRU
ECKE@F5@F9@@" :: CALL KEY(0,
U1,U2):: IF U1=15 THEN CALL
DELSPRITE(ALL):: CALL CLEAR
:: END ELSE IF U1=14 THEN RU
N ELSE 1270
1280 SUBEND
1290 SUB U(SO,SP,SQ):: IF SQ
=1 THEN 1300 ELSE DISPLAY AT
(3,12):"BEDENKZEIT[000SEK" :
: ACCEPT AT(3,23)SIZE(-3)VAL
IDATE(DIGIT):SO :: CALL HCHA
R(3,14,64,17):: SQ=1
1300 IF SO=0 THEN SUBEXIT EL
SE CALL LOAD(-31879,NULL)::
A=INT(SO/60):: B=SO-A*60
1310 CALL PEEK(-31879,TIME):
: IF TIME<47 THEN 1310 :: CA
LL LOAD(-31879,NULL):: B=B-1
 :: IF B=-1 THEN B=59 :: A=A
-1 :: IF A=-1 THEN SUBEXIT
1320 DISPLAY AT(SP*7+15,5):R
PT$("0",2-LEN(STR$(A)))&STR$
(A)&"["&RPT$("0",2-LEN(STR$(
B)))&STR$(B);
1330 CALL KEY(0,U1,U2):: IF
U2<>0 THEN 1340 ELSE IF A=0
AND B<11 THEN CALL SOUND(-50
,-1,0):: GOTO 1310 ELSE IF B
=59 THEN CALL SOUND(-50,-2,0
):: GOTO 1310 ELSE 1310
1340 SUBEND
1350 SUB Y(K,L,M,N,K$(,),ZW)
:: IF K$(K,L)="BWW" OR K$(K,
L)="BWS" OR K$(K,L)="BSS" OR
 K$(K,L)="BSW" OR K$(K,L)="P
WW" OR K$(K,L)="PSS" OR K$(K
,L)="PSW" OR K$(K,L)="PWS" T
HEN ZW=1 :: SUBEXIT
1360 A=K :: B=L :: C=M :: D=
N
1370 IF A<>C THEN A=A-1*SGN(
A-C)
1380 IF B<>D THEN B=B-1*SGN(
B-D)
1390 IF A=C AND B=D THEN ZW=
1 :: SUBEXIT ELSE IF SEG$(K$
(A,B),1,1)="F" THEN 1370 ::
IF SEG$(K$(A,B),1,1)<>"F" TH
EN ZW=0 :: SUBEXIT
1400 SUBEND
1410 SUB F1 :: CALL SOUND(20
0,220,0):: DISPLAY AT(3,12):
"FIGUR@IST@IM@WEGE" :: CALL
WAIT(250):: CALL HCHAR(3,14,
64,17):: SUBEND
1420 SUB F2 :: CALL SOUND(20
0,220,0):: DISPLAY AT(3,12):
"@@FALSCHE@FARBE@@" :: CALL
WAIT(250):: CALL HCHAR(3,14,
64,17):: SUBEND
1430 SUB Z(A,B,F$(),K$(,))::
 IF K$(A,B)="BWW" THEN CALL
D(A,B,38,142):: SUBEXIT ELSE
 IF K$(A,B)="BWS" THEN CALL
D(A,B,36,140):: SUBEXIT ELSE
 IF K$(A,B)="BSS" THEN CALL
D(A,B,32,136):: SUBEXIT
1440 IF K$(A,B)="BSW" THEN C
ALL D(A,B,34,138):: SUBEXIT
ELSE IF K$(A,B)="TWS" THEN C
ALL D(A,B,116,132):: SUBEXIT
 ELSE IF K$(A,B)="TWW" THEN
CALL D(A,B,118,134):: SUBEXI
T
1450 IF K$(A,B)="TSW" THEN C
ALL D(A,B,114,130):: SUBEXIT
 ELSE IF K$(A,B)="TSS" THEN
CALL D(A,B,112,128):: SUBEXI
T ELSE IF K$(A,B)="PWW" THEN
 CALL D(A,B,126,142):: SUBEX
IT
1460 IF K$(A,B)="PWS" THEN C
ALL D(A,B,124,140):: SUBEXIT
 ELSE IF K$(A,B)="PSS" THEN
CALL D(A,B,120,136):: SUBEXI
T ELSE IF K$(A,B)="PSW" THEN
 CALL D(A,B,122,138):: SUBEX
IT
1470 IF K$(A,B)="LWS" THEN C
ALL D(A,B,44,140):: SUBEXIT
ELSE IF K$(A,B)="LWW" THEN C
ALL D(A,B,46,142):: SUBEXIT
ELSE IF K$(A,B)="LSW" THEN C
ALL D(A,B,42,138):: SUBEXIT
1480 IF K$(A,B)="LSS" THEN C
ALL D(A,B,40,136):: SUBEXIT
ELSE IF K$(A,B)="DWW" THEN C
ALL S(A,B,2,104,6,F$(3),K$(,
)):: SUBEXIT ELSE IF K$(A,B)
="DWS" THEN CALL S(A,B,16,10
4,6,F$(2),K$(,)):: SUBEXIT
1490 IF K$(A,B)="KWS" THEN C
ALL S(A,B,16,92,5,F$(0),K$(,
)):: SUBEXIT ELSE IF K$(A,B)
="KWW" THEN CALL S(A,B,2,92,
5,F$(1),K$(,)):: SUBEXIT
1500 IF K$(A,B)="DSS" THEN C
ALL S(A,B,16,108,4,F$(3),K$(
,)):: SUBEXIT ELSE IF K$(A,B
)="DSW" THEN CALL S(A,B,2,10
8,4,F$(2),K$(,)):: SUBEXIT
```

```
1510 IF K$(A,B)="KSW" THEN C
ALL S(A,B,2,60,3,F$(0),K$(,)
):: SUBEXIT ELSE IF K$(A,B)=
"KSS" THEN CALL S(A,B,16,60,
3,F$(1),K$(,))
1520 SUBEND
1530 SUB J(A):: DISPLAY AT(2
,26):RPT$("0",3-LEN(STR$(A))
)&STR$(A):: SUBEND
1540 SUB X(Z$(,),SP):: FOR I
=1 TO 7 :: FOR II=1 TO 4 ::
CALL HCHAR(I+1,SP*6+1+II,ASC
(SEG$(Z$(SP,I),II,1))):: NEX
T II :: NEXT I :: SUBEND
1550 SUB WAIT(A):: FOR B=1 T
O A :: CALL KEY(0,C,D):: IF
D=0 THEN 1560 ELSE SUBEXIT
1560 NEXT B :: SUBEND
```


Achtung!

Neue Zeiten für unsere Hotline!

TI 99/4 A:
Montag, 15 - 19 Uhr
Tel. 089 / 129 80 13

Atari XE/XL:
Dienstag, 15 - 19 Uhr
Tel. 089 / 18 40 24

MSX I und II:
Dienstag, 16 - 19 Uhr
Tel. 089 / 129 80 13

SCHACH :

Es handelt sich um ein Schachprogramm für 2 Spieler. Mit diesem Programm können Spielstände gespeichert und wieder eingelesen werden, so daß eine vorher abgebrochene Partie beendet werden kann. Beide Spieler haben die Möglichkeit einer Rochade. Zu Beginn des Spieles müssen die Namen der beiden Spieler eingegeben werden. Daraufhin wird das Zuglimit eingegeben, wobei bei einer Eingabe von 0 kein Zuglimit existiert. Bei einem Zuglimit bricht das Programm die Partie beim Erreichen des Zuglimits ab. Danach kann zwischen dem Joystick und der Tastatur als Eingabemedium entschieden werden. Die Spieler können unterschiedliche Eingabemedien wählen. Mit der vorläufig letzten Eingabe wird die einem Spieler zur Verfügung stehende Bedenkzeit eingegeben. Bei einer Eingabe von 0 existiert keine Bedenkzeit. Daraufhin beginnt die Partie, wobei Weiß beginnt. Eine evtl. laufende Bedenkzeit kann jederzeit durch einen Tastendruck aufgehoben werden und es muß danach die Zugeingabe erfolgen. Die letzten 10 Sekunden der Bedenkzeit werden akkustisch angezeigt. Bei der Tastatureingabe müssen die Koordinaten der Spielfiguren eingegeben werden. Zuerst wird der Standort der auserwählten Spielfigur eingegeben, z.B. A2 – kein Enter –, dann wird das Zielfeld eingegeben, z.B. A4. Erst jetzt muß Enter gedrückt werden. Evtl. falsche Züge oder eine falsche Farbwahl wird vom Programm erkannt und es wird eine betreffende Fehlermeldung angezeigt. Bei derJoystickeingabe muß zuerst die gewünschte Spielfigur mit Hilfe des blauen Vierecks umrandet werden und daraufhin der Feuerknopf gedrückt werden. Das Programm markiert diese Stelle mit einem hellblauen Viereck und das Zielfeld kann auf die selbe Art ermittelt werden. Um einen Befehl einzugeben, muß entweder T oder P gedrückt werden (Joystick 1/2). Eine Rochade wird durchgeführt, wenn man ROCH eingibt. Falls dies noch nicht möglich ist oder schon durchgeführt wurde, erkennt dies das Programm und es wird eine entsprechende Fehlermeldung angezeigt. Mit der Eingabe von SAVE oder LOAD können Spielstände gespeichert und geladen werden. Bei beiden Vorgängen muß zuerst die Filename eingegeben werden. Auftretende Fehler werden hierbei abgefangen und der Filename muß erneut eingegeben werden. Die Partie wird vom Programm abgebrochen, wenn das Zuglimit erreicht wurde, ENDE eingegeben wurde oder derKönig bei einem der Spieler geschlagen wurde. Bei letzteren wird der Sieger angezeigt. Das Betätigen von Begin führt zu einem Neustart. Back beendet das Programm. Das Programm verfügt auch über die Möglichkeit, einen Bauern in einen Turm, oder in ein Pferd, oder in einen Läufer oder in die Dame zu 'verwandeln'. Dies geht aber nur, wenn der Bauer die gegnerische Grundlinie erreicht hat und von den oben genannten Spielfiguren eine fehlt (oder beide). Damit man auch immer über die letzten Züge informiert ist, erscheinen diese getrennt für Weiß und Schwarz in Feldern. Die letzten 7 Züge beider Spieler werden angezeigt. Auch die momentane Zugzahl wird angezeigt. Derjenige Spieler ist momentan dran, bei dem hinter seinem Namen ein nach links zeigender Pfeil steht und bei dem evtl. die Bedenkzeit läuft oder der Cursor zur Zugeingabe wartet.

LÖSUNGSHILFEN

Gewusst wie!

Die Scott-Adams-Adventures sind bekannt für knifflige Probleme und komplizierte Lösungen. Zwei dieser harten Nüsse haben wir für Sie geknackt.

Heute begeben wir uns bei unserem Streifzug durch die Welt der Scott-Adams-Adventures in ein Schloß, in dem magische Kräfte und verzauberte Gegenstände eine wichtige Rolle spielen.
Auf Voodoo Castle, so der Name des Schlosses, ruht ein Fluch: Der frühere Eigentümer, Count Cristo, findet keine Ruhe.
Nur Sie und der TI kommen gerade zur rechten Zeit, um ihn zu erlösen. Die Hindernisse, die sich Ihnen in den Weg stellen, werden mit unseren Lösungshilfen ein wenig einfacher zu bezwingen sein.

INHALT DER RÄUME

Es sind nur die für die Lösung wichtigen Gegenstände aufgeführt. Gegenstände, die erst sichtbar werden, wenn man eine bestimmte Handlung durchgeführt hat, stehen in Klammern.

Chapel:	(Count Cristo)
	(Ring)
Tunnel:	Knife
Chute:	Plaque
Mediums Room:	Crystal Ball
	Medium Maegen
Voodoo Room:	Page Of Voodoo Book
Window Ledge:	Doll
Fireplace:	Idol
Chimney2:	(Paper)
Stairwell:	Broken Glass
Parlor:	JuJu-Man Statue
Armory:	Sword
	Shield
Graveyard:	Clover
	Saw
Room4:	(Safe)
	(Hammer)
Lab:	Chemicals
	JuJu-Bag
	(Voodoo Book)
	(Stick)
Hole:	Rabbits Foot

EINIGE TIPS

Die Tür im Tunnel läßt sich mit einem Ring, den Sie im Sarg in der Kapelle finden, öffnen. Dazu empfiehlt es sich, den Ring genau zu inspizieren.

Der in der Feuerstelle gefundene Gegenstand beginnt zu leuchten, wenn Sie ihn säubern (DUST IDOL).
Das Labor hat zwar seine Tücken, die dort zu findenden Dinge sind aber extrem nützlich. Gegen explodierende Reagenzgläser können Sie sich schützen, wenn Sie in der Waffenkammer nachschauen. Die Chemikalien ergeben gemixt einen lekkeren Drink, der Ihnen völlig neue Wege, etwa zum Friedhof, öffnet (dies ist durchaus doppeldeutig zu verstehen). Die Tasche aus dem Labor können Sie erst dann untersuchen, wenn Sie Ihnen gehört.
Im Raum Nummer 4 läßt sich der Wandschmuck entfernen, aber nicht mit bloßen Händen. Ein Blick in die Waffenkammer hilft weiter. Die Kombination des Safes erfahren Sie, wenn Sie die im Tunnel gefundene Platte lesen. Dies funktioniert nur an einem schummrigen Platz (Fireplace oder Chimney1), wenn Sie die im Treppenhaus gefundenen Glasscherben zu Hilfe nehmen.
Der Zugang zum weiter oben gelegenen Teil des Kamins (Chimney2) ist nur zu finden, wenn Sie im unteren Teil (Chimney1) einiges abmontieren und einen Knopf drükken, der dort zum Vorschein kommt. Sie benötigen auf jeden Fall Hammer und Säge, ein HELP hilft im Kamin weiter. Sind Sie im oberen Teil angelangt, sollten Sie den Schornsteinfeger mit PUSH zum Kamin hinausbefördern – Sie erhalten dann eine interessante Information.
Die Statue des JuJu-Mannes erwacht zum Leben, wenn Sie ein magisches Wort gebrauchen. Sie sollten sich, bevor Sie die Flucht ergreifen, anhören, was Ihnen der JuJu-Mann zu sagen hat.
Der Zugang zur Grube (Hole) öff-

Voodoo Castle:

```
                +-----------+
                ! Chimney2  !
                +-----------+
                      !
                      !
                +-----------+   +---------------+
                ! Chimney1  !   ! Window Ledge  !
                +-----------+   +---------------+
                      !                !
                      !             +--+
                      !             !
                +-----------+   +---------+              +-------------+
                ! Fireplace !   ! Room 1  !              ! Voodoo Room !
                +-----------+   +---------+              +-------------+
                      !              !                         !
                      !              !                         !
                +-----------+   +---------+   +---------+   +---------+   +---------+
                ! Ballroom  !---! Chapel  !---! Tunnel  !---! Chute   !---! Medium  !
                +-----------+   +---------+   +---------+   +---------+   +---------+
                                     !
                                     !
            +---------+              !            +---------+   +---------+   +-----+
            ! Parlor  !              !            ! Room 4  !---! Room 5  !---! Lab !
            +---------+              !            +---------+   +---------+   +-----+
                  \                  !                 !
                   \                 !                 !
+----------+   +----------+   +-------------+   +---------+
! Jail Cell!---! Dungeon  !---! Stariwell   !---! Room 3  !
+----------+   +----------+   +-------------+   +---------+
                    !                !                \
                    !                !                 \
           +--------+           +---------+        +-------+
           !                    ! Room 2  !        ! Hole  !
           !                    +---------+        +-------+
           !
     +-----------------+    +---------+
  +--! Torture Chamber !----! Armory  !
  !  +-----------------+    +---------+
  !
  +---------------+
                  !
  +----------+    !
  ! Grave Yard !--+
  +----------+
```

Der Lageplan zum Text-Adventure "Voodoo Castle". Mit dieser Karte und den nützlichen Hinweisen wird die Lösung des Abenteueres zum Kinderspiel.

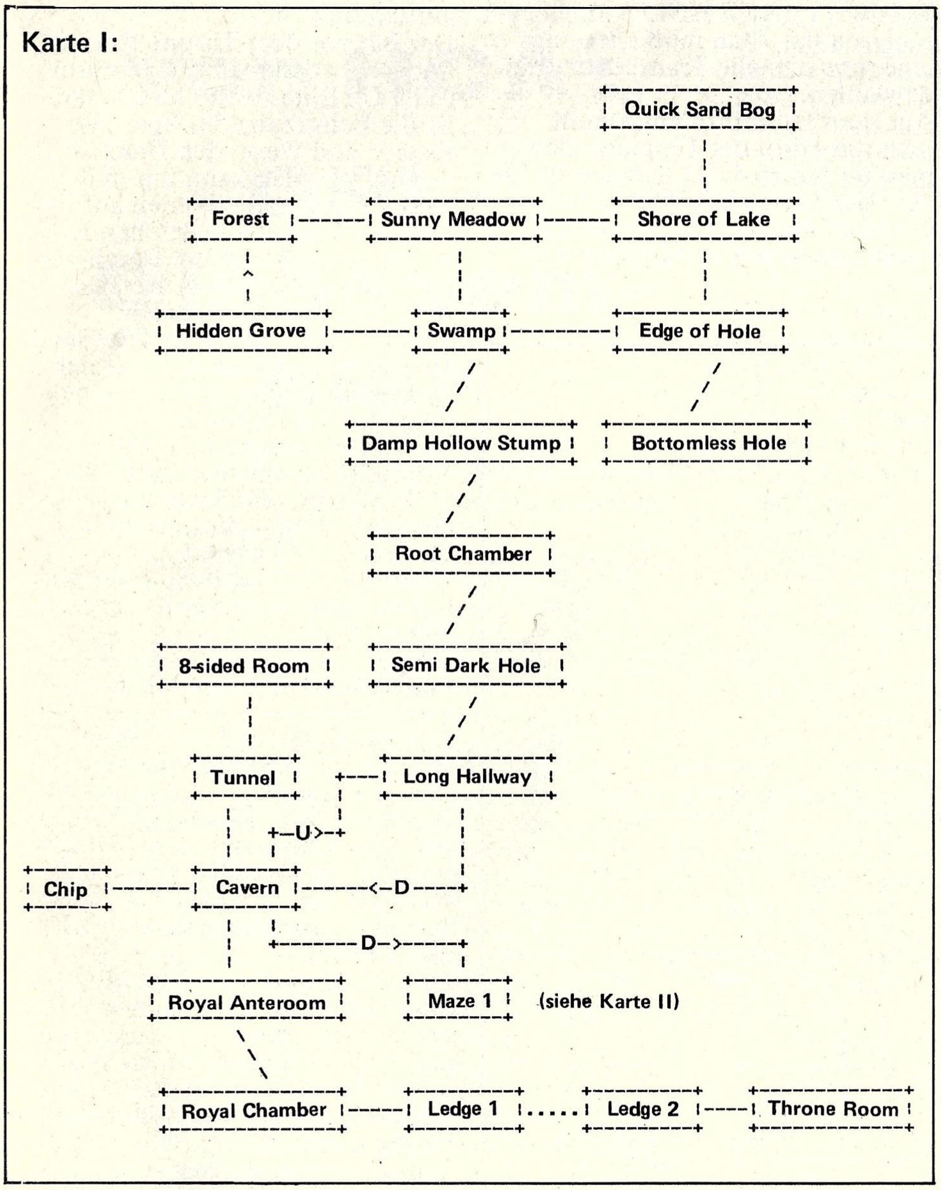
net sich, wenn Sie den Kessel im Raum Nummer 3 beiseite schieben. Sie finden dort eine Hasenpfote. Einen weiteren Glücksbringer entdecken Sie auf dem Friedhof. Tragen Sie ihn bei sich, wenn Sie im Raum 1 das Fenster öffnen wollen – er wirkt Wunder.
Wichtige Informationen über Ihren Auftrag erhalten Sie vom Medium Maegen, das Sie im Mediums Room rufen können (YELL MEDIUM). Weitere Informationen enthalten das Voodoo-Buch und die fehlende Seite des Voodoo-Buches.
Mit diesen Hilfen können Sie den Fluch, der auf Count Cristo liegt, brechen. Legen Sie dazu die Hasenpfote in den Sarg und befolgen Sie die Informationen.
Zum Öffnen des Voodoo-Room benötigen Sie die Tasche des JuJu-Mannes, welche Sie ähnlich wie den Ring gebrauchen sollten.

Wenn Sie den Keller nicht mehr verlassen können, werfen Sie einen Blick in die Kristallkugel.

GEBRAUCH DER KARTE

Auf der *Karte* sind die Verbindungen der Räume untereinander zu sehen. Zu beachten ist, daß alle Verbindungen den Himmelsrichtungen wie folgt zugeordnet sind: rechts = EAST, links = WEST, oben = NORTH, unten = SOUTH. Schräge Verbindungslinien bedeuten UP und DOWN.
Manche Verbindungen sind nicht direkt durch die Kommandos GO EAST, GO UP und so weiter anzusprechen, sondern beispielsweise durch CLIMB oder JUMP. Oft hilft auch ein GO DOOR oder GO ROOM weiter, wenn die im Plan eingezeichnete Verbindung im Spiel nicht direkt als Ausgang angezeigt wird.
Manche Zugänge öffnen sich erst, wenn bestimmte Aktionen durchgeführt worden sind. Siehe dazu die einzelnen Tips.

Mathias Eichhorn
TI-Workshop Wiesbaden□



Die Übersichts-Karte zum Spiel "Adventureland".

Hier eine weitere Folge der Lösungshilfen für die Scott-Adams-Adventures. "Adventureland" ist das erste von mittlerweile 17 bei Scott Adams erschienenen Spielen für das Adventure-Modul.

Die Aufgabe des Spielers in diesem schon fast klassisch zu nennenden Abenteuer ist es, in einer mit merkwürdigen Tieren und magischen Gegenständen angefüllten Welt nach 13 Schätzen zu suchen und sie an dem dafür vorgesehenen Ort in Sicherheit zu bringen.
Die folgenden Hinweise, eine Karte und eine Liste der für die Lösung wichtigen Gegenstände erleichtern diese schwierige Aufgabe hoffentlich ein wenig.

INHALT DER RÄUME

Es sind nur die für die Lösung wichtigen Gegenstände aufgeführt. Gegenstände, die erst sichtbar werden, wenn man eine bestimmte Handlung durchgeführt hat, stehen in Klammern.
Schätze sind durch einen vorangestellten Stern (*) gekennzeichnet.

Quick Sand Bog:	*Blue Ox
Sunny Meadow:	(*Dragon Eggs)
Shore Of Lake:	Rusty Axe *Golden Fish
Hidden Grove:	*Jewelled Fruit
Swamp:	Swamp Gas Mud Skeleton Keys
Bottomless Hole:	Flint & Steel
Damp Hollow Stump:	Bottle With Water Brass Lamp
Root Chamber:	*Pot Of Rubies
8-sided Room:	*Royal Honey Large African Bees
Royal Anteroom:	Empty Wine Bladder
Ledge 2:	*Magic Mirror
Throne Room:	*Gold Crown
Maze 5:	*Persian Rug
Bottom Of Chasm:	*Golden Net (*Firestone).

EINIGE TIPS

HELP hilft bei diesem Adventure wirklich sehr oft weiter.
Der Zugang in die Unterwelt und zum Ablageort der Schätze wird sichtbar, wenn man den im Sumpf befindlichen Baum fällt. Vorher sollte man hinaufklettern, es findet sich dort etwas Wichtiges zum Öffnen von Türen.
Zweimaliges Reiben an der Lampe (RUB LAMP) hilft weiter, wenn man zwei Schätze nicht finden sollte. Weiteres Reiben kann gefährlich sein.
Im Bottomless Hole findet man einen Feuerstein, der zum Anzünden der Lampe dient.
Zum Transport des Fischs ist ein Netz nötig, außerdem Wasser zum Feuchthalten.
Der Drache auf der Lichtung reagiert allergisch auf den Geruch von Schlamm, während die Bienen davon betäubt werden. Der Schlamm ist auch recht nützlich, wenn man im Sumpf Verletzungen erleidet.
Das Austrocknen des Schlamms während des Transports wird durch das Mitführen von Wasser verhindert.
Die Bienen kann man in der Flasche transportieren, nachdem man ein Luftloch (MAKE HOLE) hineingeschlagen hat. Man muß allerdings eine sehr schnelle Transportmethode wählen.

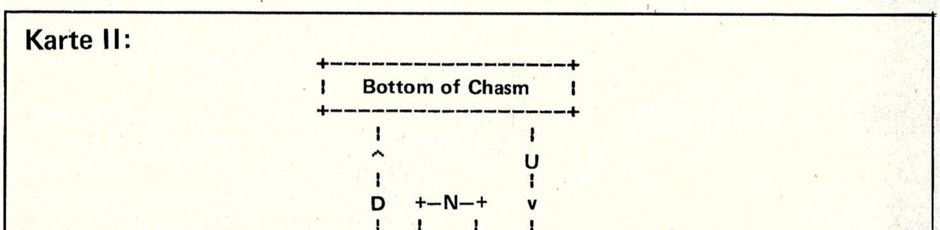
FLIEGENDER TEPPICH

Aus dem Labyrinth entkommt man mit Hilfe des Teppichs, den man im Raum 5 des Labyrinths findet. Der Teppich verfügt nämlich über die wundersame Eigenschaft, fliegen zu können. Aktiviert wird diese Fähigkeit mit SAY AWAY, was solange wiederholt wird, bis man sich wieder auf der Lichtung befindet.
In der Royal Chamber muß eine Mauer gesprengt werden, um den Zugang zum Throne Room zu finden. Eine wirkungsvolle Sprengladung erhält man, wenn man Gas in der leeren Blase, die man im Royal Anteroom findet, sammelt. Zünden kann man die Ladung mit dem Feuerstein.
Der Lavastrom läßt sich mit Hilfe von Ziegelsteinen (DAM LAVA) aufhalten.
Der Bär vor dem Throne Room flüchtet, wenn man ihn anschreit (YELL). Eine zweite Möglichkeit ist die Benutzung des Spiegels.
Es gibt drei Wege, den Drachen zu vertreiben: Man kann ihn in die Luft sprengen, die Bienen auf ihn loslassen oder die magischen Kräfte des Spiegels nutzen. Den Spiegel sollte nur auf einer weichen Unterlage abgelegt werden.
Um die Statue aus dem Treibsand zu holen, muß man sich mit der Axt hineinwagen, die Statue nehmen und das magische Wort BUNYON aussprechen. Danach empfiehlt es sich, bei Paul's Place vorbeizuschauen. Schwimmbewegungen helfen weiter, wenn man im Treibsand festsitzt.
Glühende Gegenstände lassen sich – wer hätte es erraten? – mit Wasser kühlen.

GEBRAUCH DER KARTE

Auf der Karte sind die Verbindungen der Räume untereinander zu sehen. Zu beachten ist, daß alle Verbindungen den Himmelsrichtungen wie folgt zugeordnet sind: Rechts = EAST, links = WEST, oben = NORTH, unten = SOUTH. Schräge Verbindungslinien bedeuten UP und DOWN.
Manchmal muß von dieser Regel abgewichen werden, dann sind die entsprechenden Himmelsrichtungen auf den Verbindungslinien vermerkt. Wege, die nur in einer Richtung gehbar sind, werden durch Pfeile eindeutig markiert.
Manche Verbindungen sind nicht direkt durch die Kommandos GO EAST, GO UP undsoweiter anzusprechen, sondern beispielsweise durch CLIMB oder JUMP. Oft hilft auch ein GO DOOR oder GO ROOM weiter, wenn die im Plan eingezeichnete Verbindung im Spiel nicht direkt als Ausgang angezeigt wird.

Mathias Eichhorn
TI-Workshop Wiesbaden



Diese Karte zeigt einen Ausschnitt der großen Übersichtskarte von Seite 53. Im Artikel ist eine genaue Anleitung zum Gebrauch der Karte enthalten. Auch dieses Abenteuer-Spiel kann mit den Hilfestellungen ohne Schwierigkeiten gelöst werden.

Kostenlos auf der TI-Diskette von HCA: GPL-Disassembler

Nachdem die Geheimnisse von GPL dank der Arbeit von Heiner Martin und einigen anderen Experten mittlerweile gelüftet sind, ist diese Programmiersprache auch für eigene Programmentwicklungen einsetzbar.
Notwendig ist dazu neben spezieller Hardware wie einer GRAM-Karte oder – billiger – einem GRAM-Modul auch ein Programmentwicklungssystem, welches zumindest aus Assembler, Disassembler und Debugger bestehen sollte, um vernünftig arbeiten zu können.
Als Debugger (Fehlersuchprogramm) ist zum Beispiel der mit dem Editor/Assembler-Modul gelieferte Debug oder – besser – der Explorer von Miller Graphics zu verwenden; ein Assembler wird von M. Weiand angeboten.
Der im folgenden vorgestellte GPL-Disassembler ist die Eigenentwicklung eines Mitarbeiters des TI-Workshops Wiesbaden. Er ist vollständig in Assembler geschrieben und besticht durch hohe Verarbeitungsgeschwindigkeit.
An Hardware benötigt man auf jeden Fall die 32-KByte-Erweiterung und ein Diskettenlaufwerk; ein Drucker ist empfehlenswert.
Zum Laden und Starten dient ein beliebiger Loader, der Tagged-Code in unkomprimierter Form verarbeiten kann (wie E/A, Minimem, XBasic, Supermodul des TI-Workshop Wiesbaden). Welches dieser Module man verwendet, ist völlig egal, da der GPL-Disassembler modulunabhängig geschrieben ist.
Diese Eigenschaft ist vor allem dann wichtig, wenn man die GROMs vorhandener Module disassemblieren will und dazu das zum Laden und Starten verwendete Modul aus der Konsole entfernen muß.
Nach dem Laden des Programms erscheint das Titelbild, nach Drücken einer beliebigen Taste startet das Programm und erwartet eine Eingabe.
Während der Eingabe eines Befehls werden die Tasten Delete, Insert, Erase, Left und Right abgefragt – fehlerhafte Eingaben können also leicht korrigiert werden. Enter schließt die Eingabe ab.
Es sind zwei Ausgabeformate möglich: Disassemblieren und Hexdump. Beide werden im 40-Zeichen-Modus in weißer Schrift auf blauem Hintergrund angezeigt. 22 Zeilen stehen für die Ausgabe zur Verfügung.
Beim Disassemblieren wird ein Format erzeugt, welches dem im Buch von Heiner Martin (TI 99/4A intern) verwendeten Format entspricht. Erwähnenswert ist, daß auch der FMT-Befehl nebst Unterbefehlen richtig verarbeitet wird.
Im Hexdump-Modus werden jeweils acht Byte in einer Zeile (HEX und ASCII gleichzeitig) angezeigt.
Sobald die zur Verfügung stehenden Zeilen gefüllt sind, wird der Bildschirm gelöscht und die nächsten 22 Zeilen werden ausgegeben.
Ein Unterbrechen der Bildschirmausgabe ist jederzeit durch Drücken einer beliebigen Taste möglich (was das Programm durch 'Waiting...' anzeigt). Erneutes Drükken einer Taste setzt die Ausgabe fort, während FCTN 9 die Ausführung abbricht und die Eingabe eines neuen Befehls erwartet.
Die Ausgabe kann parallel zum Bildschirm auf ein beliebiges File (wie Drucker, Disk, Ramdisk) im Format DIS/VAR 80 erfolgen.
Soweit die eher „normalen" Fähigkeiten des Programms. Etwas ungewöhnlicher ist die Möglichkeit, den zu disassemblierenden GPL-Code nicht nur aus einem GROM oder GRAM zu lesen, er kann wahlweise auch im ROM/RAM oder im VDP-RAM liegen.
Ein im Programm implementierter Loader für Files im PROGRAM-Format ermöglicht es, GPL-Code von Diskette in jeden beliebigen Speicherbereich zu laden.
Beim anschließenden Disassemblieren wird dieser Code dann so behandelt, als läge er in einem beliebigen GROM.
Der GPL-Disassembler ist im TI-Workshop Wiesbaden in regem Gebrauch und hat seine Leistungsfähigkeit bewiesen. Zu erhalten ist das Programm nebst Anleitung direkt beim Autor:
Mathias Eichhorn
Goethestraße 7
6242 Kronberg/Ts.
Es werden nur Anfragen bearbeitet, denen eine Leerdiskette und ein ausreichend frankierter Rückumschlag beigefügt sind. Eine „Bearbeitungsgebühr" oder ähnliches wird nicht erhoben.

Mathias Eichhorn
TI-Workshop Wiesbaden □

CLUB-NEWS

Hallo TI-User,
der TI-99/4A USER CLUB BERLIN lädt ein zum 1. regionalen TI-99/4A-User-Treffen in das Jugenddorf Berlin, 1000 Berlin 30, Am Karlsbad 16 am Samstag, den 30. April 1988 von 14 bis 18 Uhr.
Themen: Neuheiten, Hard- und Software, Freeware, Computerbörse, Erfahrungsaustausch, Clubarbeit, Zeitung, Bezugsquellen und Händler. Getränke und Imbiß zum Selbstkostenpreis.
Weitere Informationen entweder über die JACOBI Mailbox 030-2118390, Seite 190 oder über Kontaktadresse: Franz Neudert, Meinekestraße 25, 1000 Berlin 15, Tel.: 030/8817481 abends. □

```
10 ! ***********************
11 ! *                     *
12 ! *       ANTIPOKE      *
13 ! *                     *
14 ! *    Copyright by     *
15 ! *  WINFRIED WINKLER   *
16 ! *                     *
17 ! *                     *
18 ! *                     *
19 ! * Benoetigte Geraete  *
20 ! *  TI99/4A  Konsole   *
21 ! *     Ext. Basic      *
22 ! *  32KB-Erweiterung   *
23 ! * Disketten-Laufwerk  *
24 ! *                     *
25 ! *                     *
26 ! *  Speicherbelegung   *
27 ! *     3542 Bytes      *
28 ! *                     *
29 ! ***********************
100 DEF P(X)=X-65536*(X<0)
110 DEF HB$(X)=SEG$(H$,INT(X
/16)+1,1)&SEG$(H$,X-INT(X/16
)*16+1,1)
120 DEF HW$(X)=HB$(INT(X/256
))&HB$(X-INT(X/256)*256)
130 H$="0123456789ABCDEF->"
:: @$,D$,W$="" :: A,B,C,D,I,
J,K,L,N,W=0 :: GOTO 160
140 IMAGE ##################
############################
######################## -
- ####
150 CALL INIT :: CALL LOAD :
: CALL SCREEN :: CALL CHAR :
: CALL SOUND :: CALL HCHAR :
: CALL KEY :: CALL PEEK !@P-
160 ON WARNING NEXT :: CALL
SCREEN(4):: @$="1" :: CALL P
EEK(8198,I):: IF I<>170 THEN
 CALL INIT
170 CALL LOAD(-31806,16):: C
ALL CHAR(64,"3C4299A1A199423
C",95,"0000FFFF")
180 W$=RPT$("_",28)
190 DISPLAY AT(1,2)ERASE ALL
:"Ext.Basic 'DIS/FIX-80'SAVE
":W$:" generates 'AORG'Objec
tcode" :: DISPLAY AT(23,1):W
$:"  @ 1986 Winfried Winkler
"
200 DISPLAY AT(10,3):"1. Sho
w ""DEF-Table""": :"  2. Obj
ectcode Output": :"  3. Exit
 & New": : :"    * Your Choic
e ?  >";@$
210 ACCEPT AT(17,22)SIZE(-1)
VALIDATE("123")BEEP:W
220 IF W=3 THEN CALL HCHAR(1
,1,32,768):: CALL LOAD(-3195
2,255,231,255,231,"",-31931,
0,0)
230 CALL HCHAR(3,2,32,30)::
CALL HCHAR(10,4,32,250):: CA
LL PEEK(8194,A,B,C,D):: L=A*
256+B-2 :: N=C*256+D
240 IF N<16384 THEN 270 ELSE
 CALL SOUND(190,-3,0,110,1):
: @$="3"
250 DISPLAY AT(10,1):"* I do
n't find any Name": :"  in t
he DEF-Table !!": : :"* Pres
s any Key to Abort": :"
 or <Proceed> ??"
260 CALL KEY(0,A,B):: IF NOT
 B THEN 260 ELSE IF A<>12 TH
EN 180 ELSE CALL HCHAR(10,3,
32,250):: GOTO 360
270 IF W=2 THEN 360 ELSE J=8
280 C=(16384-N)/8 :: DISPLAY
 AT(5,3):C;"Names in DEF-Tab
le:":"   ";RPT$("-",24)
290 FOR A=N TO 16383 STEP 8
300 CALL NAME(A,W$,B):: W$="
"""&W$&""""&"  at  >"&HW$(P(
B)):: DISPLAY AT(J,5):W$ ::
C=C-1 :: J=J+1 :: IF J<19 OR
 C=0 THEN 330
310 DISPLAY AT(20,2)BEEP:"Pr
ess any Key to Continue"
320 CALL KEY(0,B,D):: IF NOT
 D THEN 320 ELSE CALL HCHAR(
8,5,32,410):: J=8
330 NEXT A
340 DISPLAY AT(J+1-(J<19),3)
BEEP:"* Done - Press any Key
 !" :: @$="2"
350 CALL KEY(0,A,B):: IF NOT
 B THEN 350 ELSE 180
360 DISPLAY AT(5,3):"Start A
ddress  =   >24F4": :"   las
t Address  =   >";HW$(P(L)):
 :"  with: Checksum ?   YES"
370 DISPLAY AT(11,6):"Line N
umber ?   YES": : :"  Output
 = DSK1.": : :"Objectcode-""
IDT"" :": :"REM Text at File
 <EOF> :": :""
380 ACCEPT AT(5,22)SIZE(-6)V
ALIDATE(H$)BEEP:W$ :: CALL D
EZ(W$,K):: ACCEPT AT(7,22)SI
ZE(-6)VALIDATE(H$)BEEP:W$ ::
 CALL DEZ(W$,L)
390 ACCEPT AT(9,22)SIZE(-3)V
ALIDATE(" YSONE")BEEP:W$ ::
C=(W$="YES"):: ACCEPT AT(11,
22)SIZE(-3)VALIDATE(" YSONE"
)BEEP:W$ :: D=(W$="YES")
400 ACCEPT AT(14,15)SIZE(-12
)BEEP:@$ :: @$="DSK"&@$ :: C
```

```
ALL HCHAR(15,11,32,16)
410 ON ERROR 560 :: OPEN #1:
@$,SEQUENTIAL,DISPLAY ,FIXED
 80,OUTPUT :: ON ERROR STOP
420 ACCEPT AT(17,21)SIZE(-8)
BEEP:W$ :: ACCEPT AT(21,1)BE
EP:@$ :: DISPLAY AT(19,1):""
:"* Please wait ...":"":""
430 I=K :: W=1 :: D$="00000"
&SEG$(W$&"        ",1,8)&"9"
&HW$(P(I))
440 DISPLAY AT(20,25)BEEP:US
ING "####":W
450 CALL PEEK(I,A,B):: D$=D$
&"B"&HB$(A)&HB$(B):: I=I+2 :
: IF LEN(D$)>59 THEN 470
460 IF I<=L THEN 450
470 GOSUB 570 :: D$="9"&HW$(
P(I)):: IF I<=L THEN 440 ELS
E D$=""
480 IF N=16384 THEN 540 ELSE
 DISPLAY AT(20,1):"   * ""DEF
"" Program :"
490 FOR A=N TO 16383 STEP 8
500 CALL NAME(A,W$,B):: DISP
LAY AT(20,22):W$ :: ACCEPT A
T(20,22)SIZE(-6)VALIDATE(UAL
PHA,DIGIT,"#$%&()+-")BEEP:W$
510 IF W$="" THEN 520 :: D$=
D$&"6"&HW$(P(B))&W$&RPT$(" "
,6-LEN(W$)):: IF LEN(D$)>54
THEN GOSUB 570 :: D$=""
520 NEXT A
530 IF D$<>"" THEN GOSUB 570
540 D$=":* EOF * ANTIPOKE *:
" :: IF @$<>"" THEN D$=D$&"
   < "&@$&" >"
550 PRINT #1:D$ :: CLOSE #1
:: @$="3" :: GOTO 180
560 CALL SOUND(190,-3,0,110,
1):: DISPLAY AT(15,10):"* Fi
le Error!" :: RETURN 400
570 IF C=0 THEN D$=D$&"80000
F" :: GOTO 590
580 B=0 :: FOR J=1 TO LEN(D$
):: B=B+ASC(SEG$(D$,J,1)):: 
NEXT J :: B=(NOT(B+55))+
65537 :: D$=D$&"7"&HW$(B)&"F
"
590 IF D THEN PRINT #1,USING
 140:D$,W ELSE PRINT #1:D$
600 W=W+1 :: RETURN !@P+
610 SUB DEZ(I$,O):: IF ASC(I
$)<>62 THEN O=VAL(I$):: SUBE
XIT
620 IF LEN(I$)<5 THEN I$=SEG
$(">0000",1,6-LEN(I$))&SEG$(
I$,2,5)
630 H$="123456789ABCDEF" ::
DEF D(X$)=POS(H$,SEG$(X$,1,1
),1)*16+POS(H$,SEG$(X$,2,1),
1)
640 O=D(SEG$(I$,2,2))*256+D(
SEG$(I$,4,2)):: O=O+65536*(O
>32767):: SUBEND
650 SUB NAME(A,N$,S):: CALL
PEEK(A,B,C,D,E,F,G,H,I):: S=
256*H+I :: N$=CHR$(B)&CHR$(C
)&CHR$(D)&CHR$(E)&CHR$(F)&CH
R$(G):: SUBEND
660 END
```

Antipoke-Anleitung:

Benötigt werden: ExBASIC, 32 KByte, Disk.
Wohl jeder TI-User hat aus den seligen Kassettenrecorder-Zeiten, oder weil Zeitschriften nicht den Quellcode abgedruckt haben, noch einige Pokelisten.
Diese sind aber, sofern man sich der üblichen Methode: READ :: CALL LOAD ... bedient, sehr langsam geladen. Außerdem beanspruchen sie zusätzlichen Speicherplatz.
Das Programm Antipoke speichert jeden genannten Speicherinhalt, in den zuvor eine Pokeliste gepoked wurde, als Object-Code DIS/FIX80 ab. Und zwar mit absoluter Ladeadresse, uncompressed.

Option #1 listet alle DEFinierten Assembler-Startnamen am Bildschirm.
Option #3 beendet das Programm und löscht ("NEW").
Option #2 saved den Object-Code; einzugeben sind:

- Start- und Endadresse (mit ">"hexadezimal oder ohne dezimal möglich). Die vom Programm vorgegebenen Werte entsprechen hier den Einträgen im REF/DEF-Table, sofern das POKE-Programm sie korrekt gesetzt hat. Ansonsten sind die End-/Start-Werte der CALL-LOAD-Schleife aus dem POKE-Programm zu verwenden.
- Filenamen des Object-Codes (Error-Check implementiert);
- Checksum Y/N: Soll der Object-Code Prüfsummen enthalten?
- Line Number Y/N: Soll der Object-Code Zeilennummern erhalten?

IDT: Hier können die acht Buchstaben für File-Identifizierung eingegeben werden, die dann (TAG#0) am Object-Code-Anfang stehen.
REM: Bis zu 28 Buchstaben, die am Ende des Object-Codes angefügt werden. Diese werden nicht geladen und benötigen auch keinen zusätzlichen Platz auf Diskette, sie haben also nur eine REM-Zeilen-Funktion.
Während der Abarbeitung des Speichers wird die laufende Zeile des Object-Codes angezeigt.
Ist der gesamte Speicher gesaved, wird der gesamte DEF-Table abgearbeitet:
Es wird jeder DEFinierte Assembler-Startname angezeigt, der verändert werden kann, und durch <Enter> in den Object-Code eingebunden.
Bei Löschen und Eingabe eines Leerstrings wird der entsprechende Startname nicht in den Object-Code übernommen. Das ist zum Beispiel der Fall, wenn er zu einem jetzt nicht abgespeicherten Programmteil gehört.
Der DEF-Table wird schon vor Beginn des SAVEns geprüft. Falls er leer ist, also der Verdacht nahe liegt, daß sinnloses Zeug abgespeichert wird, wird eine Warnung ausgegeben, die mit <Proceed> übergangen werden kann.
Nach dieser Prozedur liegt die Pokeliste als Object-Code vor und kann künftig schnell und bequem in Programme eingebunden werden.

Winfried Winkler
System 99 User Group, Berlin □

Es muss nicht immer die 232 sein

Es soll doch tatsächlich Freaks geben, die aus irgendwelchen unerfindlichen Gründen zwei TIs direkt zusammenschalten wollen. Meist tun sie es über die RS232. Es geht aber auch einfacher.
Bedingung dafür ist, daß nicht beide Computer gleichberechtigt sind, sondern der eine als Master und der andere als Slave arbeitet. Der Slave hat nur zu antworten, wenn er etwas gefragt wird, da der Master sonst Unsinn auf den Bildschirm bringt. Auch darf am Slave für die Dauer der Verbindung keine Taste gedrückt werden, sonst bekommt der Slave Unsinn über seinen Port.
Die Hardware ist denkbar einfach: Ein vier- oder fünfadriges Phonokabel mit Schirm, zwei Joystickstecker und vier Leuchtdioden sind alles, was man braucht. Damit wird ein Kabel gelötet (siehe Abbildung).

Im Gegensatz zu den Ohrhörer-Eingängen und der Motorsteuerung haben nur die Mikrofon-Stecker eine echte Masse. Die Masseleitung wird daher per Krokodilklemmen auf beiden Seiten mit den Masse-Anschlüssen der Mikrofon-Ausgänge des Recorderkabels verbunden.
Die Leuchtdioden dienen nur zur Überwachung des Datentransportes und benötigen keine Vorwiderstände. Diese sind im TI bereits enthalten.
Der Rest ist ein reines Softwareproblem, das mit Maschinenroutinen lösbar ist, die auch von BASIC her aufrufbar sind. Der Slave kann über seine CRU-Bit 4 und 5 Daten seriell empfangen.

Soll das Bit 4 des Slave gesetzt werden, so muß der Master nur die Befehle

```
I.   CLR R12
     SBO 20
     SBZ 18
     SBO 19
```

ausführen. Soll das Bit 5 des Slave aktiv werden, so lautet die Befehlsfolge

```
II.  CLR R12
     SBO 20
     SBO 18
     SBO 19
```

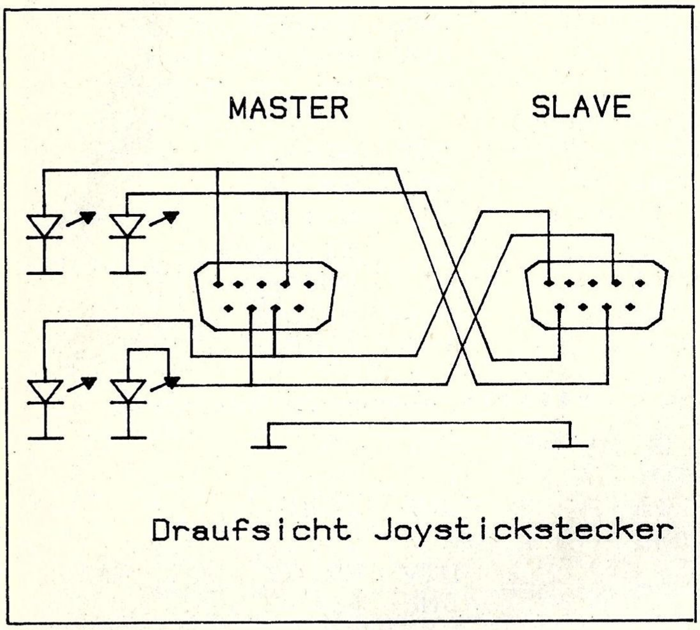
Sobald der Slave das erkennt, kann er den Empfang des Bit quittieren: Entweder arbeitet er die Befehlsfolge I ab, was im Master das CRU-Bit 4 aktiv werden läßt. Oder er arbeitet II ab, wodurch im Master das Bit 6 (nicht 5!) aktiv wird. Der Master führt dann

```
III. CLR R12
     SBZ 20
     SBZ 19
```

durch. Er nimmt also sein Bit zurück und wartet, bis der Slave auf die Rücknahme mit der gleichen Befehlsfolge reagiert.
Damit ist ein Bit übertragen und ein neues kann folgen. Die Routinen zur Übertragung eines solchen Bit kann man auch zu einem durch CALL LINK vom BASIC her aufrufbaren Subprogramm zusammenfassen. Bei dieser sogenannten Handshake-Steuerung lassen sich problemlos 8000 Baud (Bit pro Sekunde) Datengeschwindigkeit erreichen. Wichtig auch: Der Master darf nur im Tastaturmodus 0 betrieben werden, da er sonst seine Joysticks abfragt und somit der Slave Unsinn empfängt.
Der Verbindungsaufbau geschieht wie folgt:

Der Slave wird gestartet und arbeitet die folgende Routine, die auch als ein zweites, mittels CALL LINK aufrufbares, Subprogramm gehandhabt werden kann, ab:

```
      Verbindungsaufbau
                !
                !
                V

   Slave wartet auf Anweisung
   Master setzt Bit 4: Slave soll       <-----------------------+
                       Daten empfangen                          !
   Master setzt Bit 5: Slave soll                               !
                       Daten senden                             !
                                                                !
      !                          !                              !
      ! Bit 4                    ! Bit 5                        !
      !                          !                              !
      V                          V                              !
                                                                !
Warten auf Daten             Warten auf Anweisung,              !
                             welche Daten gesendet              !
                             werden sollen                      !
                                                                !
      !                          !                              !
      !                          !                              !
      V                          V                              !
                                                                !
Daten im Speicher            Daten ausgeben                     !
ablegen                                                         !
                                                                !
      !                          !                              !
      !                          !                              !
      +--------------------------+------------------------------+
```



**Oben: Schaltplan zum Aufbau der Verbindung zwischen zwei TI-Konsolen.
Links: Protokoll der Daten-Übertragung.**

```
CLR R12
CLR R0
TB  4
JEQ $-4  Warten, bis das
DEC R0   CRU-Bit 4 vom
JNE $-6  Master gesetzt
         wird
SBO 20   Antworten, in-
SBO 18   dem das CRU-
SBO 19   Bit 6 im Master
         gesetzt wird

CLR R0
TB  4    Warten auf
JNE $-2  Rücknahme
DEC R0   des Bit 4
JNE $-6

SBZ 20   Bit 6 zurück-
         nehmen.
```

Der Slave wartet, bis der Master sich meldet. Die Meldung quittiert er mit dem Setzen des CRU-Bit 6, worauf der Master seine Meldung (das CRU-Bit 4) zurücknimmt. Sofort nimmt auch der Slave sein Bit von der Leitung, so daß die Datenübertragung beginnen kann.
Das alles können Sie an den Leuchtdioden verfolgen. Wichtig ist: Die Leuchtdioden leuchten, wenn die entsprechende Leitung *passiv* ist, und sie gehen aus, wenn die Leitung *aktiv* wird.
Die Routine, die der Master ausführen muß, wenn der Slave auf seine Meldung wartet, ist:

```
CLR R12
SBO 20
SBZ 18   Bit 4 im Slave
SBO 19   setzen

CLR R0   Warten auf Ant-
TB  6    wort des Slave
JEQ $-4
DEC R0
JNE $-6

SBZ 20   Bit 4 zurück-
         nehmen

CLR R0
TB  6
JNE $-4  Warten, bis der
DEC R0   Slave das Bit 6
JNE $-6  wieder löscht.
```

Mit Absicht wurde kein vollständiges Programm veröffentlicht. Es hängt vom Verwendungszweck ab, was Sie wie übertragen. Einen einfachen Algorithmus zum Austausch von Werten in beide Richtungen für den Slave sehen Sie in der Abbildung. Die Daten selbst werden seriell im Binärformat übertragen. Das Setzen des Bit 4 bedeutet dann eine Null und das Bit 5 beziehungsweise 6 eine Eins. *Sven Dyroff*□

Extended-Basic Merger Rev. # 2

Der Extended-BASIC-Befehl MERGE "Device.Filename" erlaubt das zeilenweise Einlesen von Programmen, sogar zusätzlich zu denen, die bereits im Speicher sind.
Jedoch wird sein Nutzen eingeschränkt durch die fehlende Möglichkeit, auch einzelne Zeilen eines größeren Programmes abzuspeichern. Es mußten alle nicht benötigten Programmzeilen gelöscht werden, bevor man das gesamte Programm mit

SAVE "Device.Filename",MERGE

absaven konnte.
Das hier vorliegende Programm, ausschließlich unter Extended BASIC (nicht Ed/As oder Minimem!) lauffähig, beseitigt diesen Mangel.
Aufruf:

CALL LINK("MERGER","Device.Filename",
Startzeile,Endzeile)

Das Programm kann nur im Direktmodus, also nicht aus einem laufenden Programm heraus, aufgerufen werden. Es ist gegen jede Fehlbedienung, wie nicht vorhandene Zeilennummern, abgesichert und gibt eine entsprechende Fehlermeldung ohne Absturz.
Ist die Start- beziehungsweise Endzeile nicht im Programm vorhanden, wird die nächstgrößere beziehungsweise kleinere Zeilennummer im angegebenen Intervall verwendet. Das heißt, daß mit

CALL LINK ("MERGER","Device.Filename",
1,32767)

wie bisher jedes beliebige Programm komplett abgespeichert werden kann; nur schneller, als dies der Extended-BASIC-Befehl täte.
Warnung:
Die einzige mögliche Software-Ursache für einen Absturz des Programms ist eine Manipulation des Zeilenlängen-Byte – eine beliebte Variante, um einen LIST-Schutz zu erreichen, etwa: Längenbyte = Null.
Seien Sie sicher, daß das Zeilenlängen-Byte für jede abzuspeichernde Zeile den korrekten Wert hat.
Anmerkung:
Ein so geschütztes Programm ist zwar immer noch fehlerfrei lauffähig, aber nicht mehr listbar. Ein Absturz erfolgt bei Länge = Null, oder, in anderen Fällen, bei unvollständigem Listing. *Wilfried Winkler*□

Copyright (C)1986:
Winfried Winkler
Rathenower Str. 69
D-1000 Berlin 21
Tel.: 030/3944749

NUTZEN SIE DEN KOSTENLOSEN ANZEIGENTEIL

```
****************************************
* LINK("MERGER","Filename",von,bis)   *
****************************************
* XBasic-Zeilenlaengen-Byte darf als *
* List-Schutz nicht geloescht sein!  *
****************************************
       RORG
       IDT   <MERGER>
       DEF   MERGER

DSRWS  BSS   32
REGIST BSS   24
DSRBUF BSS    8

PABDAT DATA >12,>620,>A300,0,0
FILNAM TEXT  DSK#.DATEI-NAME'
FILLEN BYTE 15

MERGER LWPI REGIST
       CLR   R9
       CB    @>8344,R9
       JEQ   CONT1
       LI    R0,>1A00
       JMP   ERROR

CONT1  C     @>8330,@>8332
       JNE   CONT2
       LI    R0,>1D00
       JMP   ERROR

CONT2  CLR   R0
       LI    R1,01
       LI    R2,FILNAM-1
       MOVB @FILLEN,*R2
       BLWP @>2014

       INC   R1
       BLWP @>200C
       BLWP @>2018
       DATA >12B8
       CB    @>8354,@OVF
       JEQ   BADLIN
       MOV   @>834A,R7
       JGT   CONT3

BADLIN LI    R0,>1700
ERROR  BLWP @>2034

CONT3  INC   R1
       BLWP @>200C
       BLWP @>2018
       DATA >12B8
       CB    @>8354,@OVF
       JEQ   BADLIN
       MOV   @>834A,R8
       JLT   BADLIN
       C     R8,R7
       JLE   BADLIN
       MOV   @>8332,R6
       MOV   @>8330,R5
       DEC   R6

OPEN   DEC   R2
       MOV   *R2,R2
       AI    R2,10
       LI    R1,PABDAT
       LI    R0,>600
       BLWP @>2024
       BLWP @DSR

******* suche Zeilennummern

LOOPI  DECT R6
       C     R6,R5
       JHE   CONTI

       MOV   R9,R9
       JNE   EOF
       LI    R0,>1600
       JMP   ERROR

CONTI  MOV   R6,R3
       MOVB *R3+,R1
       SWPB R1
       MOVB *R3+,R1
       SWPB R1
       C     R8,R1
       JLT   EOF
       C     R1,R7
       JHE   SAVE

NEXTI  DECT R6
       JMP   LOOPI

EOF    B     @RETURN

****** schreibe Zeile ins File

SAVE   LI    R4,>620
       BL    @>23E6

       MOVB *R3+,R1
       SWPB R1
       MOVB *R3,R1
       SWPB R1
       MOVB @>FFFF(R1),R2
       SRL   R2,8
       LI    R0,>622
       BLWP @>2024

       INCT R2
       SWPB R2
       MOVB R2,R1
       LI    R4,>605
       BL    @>241A

       LI    R4,>600
       LI    R1,>300
```

```
OVF     EQU  $-2
        BL   @>241A
        SETO R9
        BLWP @DSR
        B    @NEXTI

****** Close & Ende

RETURN  LI   R4,>600
        LI   R1,>300
        BL   @>241A
        LI   R4,>620
        SETO R1
        BL   @>23E6

        LI   R4,>605
        LI   R1,>200
        BL   @>241A
        BLWP @DSR

        LI   R4,>600
        LI   R1,>100
        BL   @>241A
        BLWP @DSR

        CLR  @>837C
        LWPI >83E0
        B    @>70

****** DSR-Aufruf

DSR     DATA DSRWS,$+2

        LI   R3,>609
        MOV  R3,@>8356
        BL   @>2406
        MOVB R1,R0
        SRL  R0,8
        SETO R4
        LI   R2,DSRBUF

DSRLN1  INC  R3
        INC  R4
        CI   R4,07
        JGT  DSRERR
        C    R4,R0
        JEQ  DSRLN2
        BL   @>2406
        MOVB R1,*R2+
        CB   R1,@>0CF7
        JNE  DSRLN1

DSRLN2  MOV  R4,R4
        JEQ  DSRERR
        CLR  @>83D0
        MOV  R4,@>8354
        INC  R4
        A    R4,@>8356

DSRLN3  LWPI >83E0
        CLR  R1
        LI   R12,>0F00

DSRLN4  MOV  R12,R12
        JEQ  DSRLN5
        SBZ  0
DSRLN5  AI   R12,>0100
        CLR  @>83D0
        CI   R12,>2000
        JEQ  DSRERR
        MOV  R12,@>83D0
        SBO  0
        LI   R2,>4000
        CB   *R2,@>000D
        JNE  DSRLN4
        AI   R2,08
        JMP  DSRLN7

DSRLN6  MOV  @>83D2,R2
        SBO  0
DSRLN7  MOV  *R2,R2
        JEQ  DSRLN4
        MOV  R2,@>83D2
        INCT R2
        MOV  *R2+,R9
        MOVB @>8355,R5
        JEQ  DSRLN9
        CB   *R2+,R5
        JNE  DSRLN6

        SRL  R5,8
        LI   R6,DSRBUF
DSRLN8  CB   *R2+,*R6+
        JNE  DSRLN6
        DEC  R5
        JNE  DSRLN8

DSRLN9  INC  R1
        BL   *R9
        JMP  DSRLN6

        SBZ  0
        LWPI DSRWS
        LI   R3,>601
        BL   @>2406
        SRL  R1,13
        JNE  DSRERR
        RTWP

DSRERR  LWPI REGIST
        MOV  @>8340,R4
        MOV  R4,@>8304
        LI   R3,>600
        BL   @>23CA
        AI   R4,04
        BL   @>23E6
        LI   R0,>2400
        BLWP @>2034

        END
```


IMPRESSUM

HCA (MSX&TI-REVUE), die Zeitschrift für die Freunde des aktiven Computerns, erscheint monatlich in der Computer Aktiv Verlags GmbH i.G.
Tel.: 089/129 80 11
Telex: 5214428 cav-d

Verantwortlich für den Inhalt:
Lukas Stock (Chefredakteur)
Hartmut König (stv. Chefred.)
Redaktion: Thomas Dehn, Sven Dyroff, Christoph Kögler, Hartmut König, Jürgen Puschmann, Uwe Schröder, Jürgen Stelter, Lukas Stock

Geschäftsführer:
Werner E. Seibt

Alle: Postfach 1161
8044 Lohhof.
Anfragen bitte nur schriftlich Es gilt die Honorarliste des Verlages.

Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung. Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für den Abdruck sowie die Aufnahme in den Programm-Service nach den Verlags-Sätzen und überträgt dem Verlag das Copyright sowie das Recht auf wirtschaftliche Verwertung. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt, Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages. Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.
Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München

Abo- und Kassetten-Service: Henny Rose Seibt

ISSN 0933-3800

©by CA Verlag, Heßstraße 90, 8000 München 40 SPS und Autoren

TIPS & TRICKS

Darf's acht Kilo mehr sein?

Durch einen Trick stehen dem BASIC-Programmierer acht KByte RAM zum Zwischenspeichern von Daten oder für eigene Maschinenprogramme zur Verfügung. Acht KByte, die kein einziges Bit des für BASIC-Programme verfügbaren RAM-Bereiche belegen. Wir zeigen Ihnen, wie es funktioniert.

Dieses „unsichtbare" RAM ist bei eingeschaltetem Atari-BASIC nicht verfügbar, läßt sich aber durch einen Trick trotzdem ansprechen. Dieser funktioniert auf allen Atari-Computern der XL- und XE-Serie, nicht jedoch auf den älteren Geräten.
Bekanntlich belegt das BASIC-ROM den Speicherbereich von $A000 bis $BFFF und benötigt somit acht KByte des Hauptspeichers.
Wird BASIC nicht gebraucht, weil gerade ein Maschinenprogramm läuft, so läßt sich das BASIC-ROM ohne größere Schwierigkeiten abschalten. Wenn Sie beim Booten einer Spieldiskette die Option-Taste drükken, dann geschieht dies ganz automatisch.
Unter dem BASIC-ROM befinden sich acht K-Byte RAM, die von vielen Maschinenprogrammen genützt werden.
Das Umschalten zwischen BASIC und RAM kann auch durch Software gelöst werden, in BASIC würde dafür ein POKE ausreichen. Nur hätte das wenig Sinn.
Der Rechner würde danach unweigerlich abstürzen, weil BASIC nach diesem Befehl ausgeschaltet ist und der Prozessor vergeblich versuchen würde, darauf zuzugreifen. Deshalb kann der unter dem BASIC-ROM verborgene RAM-Speicher nur von Maschinenprogrammen verwendet werden.

Wie bei jeder Regel, gibt es auch hier eine Ausnahme, nämlich die USR-Funktion des Atari-BASIC. Mit ihr können Maschinenprogramme von BASIC aufgerufen werden, die den RAM-Bereich unterhalb des BASIC-ROM nützen.
Dazu muß das Maschinenprogramm vor dem Zugriff auf dieses RAM des BASIC ausschalten. Es kann daraufhin Daten unter dem BASIC-ROM ablegen oder ein Unterprogramm aufrufen, das „unter" BASIC versteckt wurde.
Bevor die Kontrolle an BASIC zurückgegeben wird, muß es natürlich wieder eingeschaltet werden. Das An- und Ausschalten hat keinen Einfluß auf den Speicherinhalt des verborgenen RAM-Bereiches, es steuert lediglich den Zugriff der CPU.
Für diese Programmiertechnik gibt es eine ganze Reihe sinnvoller Anwendungen, schließlich stehen auf diese Art acht KByte Speicher für Maschinenprogramme und Daten zur Verfügung.

BASIC-SCHALTER

Stellen Sie sich folgende Situation vor: Sie haben gerade die letzte Zeile eines BASIC-Programmes eingegeben und das fertige Listing mehrmals gesichert. Zur Entspannung wollen Sie jetzt ein Spiel laden und rufen mit DOS das Disketten-Betriebssystem auf. Da fällt Ihnen ein, daß das Spiel nur bei ausgeschaltetem Atari-BASIC läuft. Normalerweise müssen Sie jetzt den Computer ausschalten und erneut booten, wobei Sie diesmal die Option-Taste beim Einschalten gedrückt halten.
De erneuten Bootvorgang können Sie sich jedoch sparen, wenn Sie unser *Listing* 'SCHALTER.BAS', verwenden.
Dieses BASIC-Programm erzeugt ein kleines Maschinenprogramm, welches unter dem Namen 'SCHALTER.COM' auf Diskette gespeichert wird. Kopieren Sie dieses File am besten auf Ihre Spiele-Disketten.
Da es nur einen Sektor belegt, paßt es auf fast jede Diskette. Dieses Programm schaltet BASIC aus, wenn es zuvor eingeschaltet war und umgekehrt.

```
10 REM **********************
11 REM * HOMECOMPUTER AKTIV *
12 REM *   PRAESENTIERT:    *
13 REM *--------------------*
14 REM *  BASIC - SCHALTER  *
15 REM *--------------------*
16 REM *   MASCHIENENCODE   *
17 REM * DATAS: ATARI BASIC *
18 REM *--------------------*
19 REM *     DATEINAME      *
20 REM * "D:SCHALTER.BAS"   *
21 REM *--------------------*
22 REM * AUTOR DES LISTINGS *
23 REM *  1987 C. KOEGLER   *
24 REM **********************
25 OPEN #1,8,0,"D:SCHALTER.COM"
26 RESTORE :CK=0:FOR S=0 TO 36:READ
WERT:CK=CK+WERT:NEXT S
27 READ WERT:IF CK<>WERT THEN ? CHR$
(253);CHR$(125);"DATENFEHLER":END
28 CLOSE #1:END
29 DATA 255,255,0,6,24,6
30 DATA 172,1,211,169,56,141,3,211,1
69,255,141,1,211,169,60,141,3,211,15
2,73,2,141,1,211,96
31 DATA 226,2,227,2,0,6,4010
```

RÜCKBLICK

Im Laufe der Jahre ist eine Reihe von Modellen der Acht-Bit-Ataris auf den Markt gekommen. Angefangen beim Atari 400 und 800, ging es über den 600 XL und den 800 XL bis hin zum 130 XE.
Als letzter wurde im Herbst 1987 der Atari 800 XE vorgestellt, der sich nur durch das neue Gehäuse vom Atari 800 XL unterscheidet.
In der Anfangszeit der Homecomputer waren RAM-Bausteine noch sehr teuer. Deshalb wurde der Atari 400 und 800 nur für maximal 48 K-Byte RAM ausgelegt. Bei diesen Rechnern war BASIC nicht eingebaut, sondern in einem ROM-Modul untergebracht, welches der Anwender zusätzlich erwerben mußte.
Als nun der Atari 800 XL und sein kleinerer Bruder, der 600 XL, auf den Markt kamen, hatten sich die Preise für Speicherbausteine erheblich ge-

senkt. Dies war die Ursache einiger Änderungen. So verfügt der Atari 800 XL über 64 KByte RAM. Dazu kommen noch 16 KByte Betriebssystem-ROM sowie BASIC, welches weitere acht KByte ROM belegt. Dies ergibt eine Gesamtsumme von 88 KByte Speicher.
Da der 6502-Microprozessor des Atari jedoch nur 64 KByte auf einmal adressieren kann, muß zwischen verschiedenen Speicherbausteinen hin- und hergeschaltet werden. Diese Technik wird in der Fachsprache als Bankswitching bezeichnet.

SPEICHER-VERWALTUNG

Die Auswahl der einzelnen Speicherbänke übernimmt ein Baustein mit der Bezeichnung MMU (Memory Management Unit, zu deutsch: Speicherverwaltungs-Einheit). Zur Steuerung dieser MMU wird der Port B der PIA verwendet, der bei den Modellen 400 und 800 zur Abfrage der Joystick-Ports 3 und 4 verwendet wurde. Deshalb verfügen die neueren Modelle nur noch über zwei Joystick-Ports.
Da das Atari-BASIC früher ein Steckmodul war, belegt es den für diese ROM-Module vorgesehenen Speicherplatz zwischen $A000 und $BFFF.

BASIC EIN- UND AUSSCHALTEN – SO WIRD'S GEMACHT

Zur Auswahl zwischen BASIC-ROM und RAM dient Bit 1 von Port B. Ist es gesetzt, so können im Bereich zwischen $A000 und $BFFF acht KByte RAM angesprochen werden. Ist dieses Bit gelöscht, so wird das BASIC-ROM eingeblendet. Durch Invertieren des Bit wird BASIC ein- oder ausgeschaltet.
Die folgenden Beispiele zeigen Ihnen, wie es gemacht wird:

1. BASIC ausschalten:

```
10 LDA $D301
20 AND #$FD
30 STA $D301
```

2. BASIC anschalten:

```
10 LDA $D301
20 ORA #$02
30 STA $D301
```

3. BASIC umdrehen:

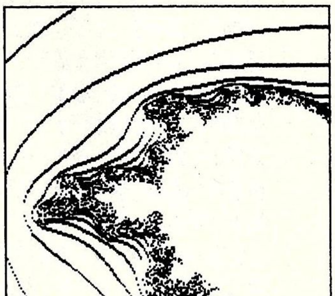
```
10 LDA $D301
20 EOR #$02
30 STA $D301
```

Die dritte Beispiel-Routine dreht den Status von BASIC um. War es angeschaltet, so wird es ausgeschaltet und umgekehrt.
Die vorgestellten Programmier-Techniken sind universell einsetzbar. In acht KByte lassen sich viele Unterprogramme oder zum Beispiel ein kompletter Graphics-8-Bildschirm verbergen. Für Zeichensätze, Player-Missile-Daten, oder gar eine Display-List ist dieser Bereich jedoch nicht verwendbar.

Christoph Kögler □

Listing zum Artikel: SCHALTER.BAS

Erzeugt ein sehr kurzes Maschinen-Programm auf Diskette. Mit diesem können Sie das BASIC jederzeit an- und ausschalten.



Space-Patrol

Bei diesem Programm muß man in einem Koordinatensystem (10x10) vier feindliche Raumschiffe suchen. Mit einem Fadenkreuz, das mit dem Joystick 1 gesteuert wird, sollen mit möglichst wenigen Versuchen diese Raumschiffe sichtbar gemacht werden. Dabei wird ein „Treffer" nur registriert (Raumschiff wird sichtbar), wenn das Fadenkreuz mit den Koordinaten eines Raumschiffes übereinstimmt und die „Feuertaste" gedrückt wurde.
Ansonsten wird durch Betätigen der Feuertaste des Joysticks nur die Anzahl der versteckten Raumschiffe angezeigt, die sich vom jeweiligen Standort aus in horizontaler, vertikaler und diagonaler Richtung befinden.
Außerdem wird an dem unteren Rand des Bildschirms noch die Zahl der Versuche angezeigt.
Im Programm selbst ist eine Spielanleitung vorhanden. □

```
100 ! **********************
110 ! *                    *
120 ! *   SPACE - PATROL   *
130 ! *                    *
140 ! *    Copyright by    *
150 ! *   Siegmar Cyrnik   *
190 ! * Benoetigte Geraete *
200 ! *  TI99/4A  Konsole  *
210 ! *     Ext. Basic     *
220 ! *      Joystick      *
260 ! *  Speicherbelegung  *
270 ! *     8104 Bytes     *
280 ! *                    *
290 ! **********************
300 CALL CLEAR :: CALL SCREE
N(12)
310 DISPLAY AT(9,8):"*******
*********" :: DISPLAY AT(13,
8):"****************"
320 DISPLAY AT(11,10):"SPACE
-PATROL"
330 CALL CHAR(120,"3C7EFFE7E
7FF7EDB")
340 FOR I=14 TO 20 STEP 2
350 CALL COLOR(12,14,12):: C
ALL HCHAR(18,I,120,1)
360 NEXT I
370 CALL CHAR(64,"3C4299A1A1
99423C")
380 DISPLAY AT(24,3):"@ 1986
 BY SIEGMAR CYRNIK"
390 FOR I=1 TO 24 :: READ DM
,FM,LM
400 CALL SOUND(DM,FM,LM):: N
EXT I
410 DATA 150,524,1,150,660,1
,300,784,0,150,698,1,150,588
,1,400,524,0,200,30000,1,150
```

```
,660,2
420 DATA 150,784,2,300,988,1
,150,880,2,150,698,2,400,660
,1,200,30000,1,200,588,1,100
,30000,1
430 DATA 400,784,0,150,30000
,1,200,660,1,150,30000,1,250
,588,1,150,30000,1,500,524,0
,100,44000,30
440 CALL CLEAR :: CALL SCREE
N(2):: FOR I=2 TO 8 :: CALL
COLOR(I,16,2):: NEXT I
450 RANDOMIZE :: CALL CHAR(4
0,"1"):: CALL CHAR(48,"101")
:: CALL CHAR(56,"00001")
460 FOR N=1 TO 25 :: CALL VC
HAR(INT(RND*23)+1,INT(RND*31
)+1,8*(INT(RND*3)+5)):: NEXT
 N
470 DISPLAY AT(3,9):"SPACE-P
ATROL" :: DISPLAY AT(5,13):"
ODER" :: DISPLAY AT(7,3):">W
ER SUCHET, DER FINDET<"
480 FOR I=13 TO 19 STEP 2 ::
 CALL COLOR(12,14,2):: CALL
HCHAR(9,I,120,1):: NEXT I
490 DISPLAY AT(16,11):"DRUEC
KE" :: DISPLAY AT(18,6):"1 F
UER ERKLAERUNG" :: DISPLAY A
T(20,6):"2 FUER SPIELANFANG"
500 DISPLAY AT(24,1):">ALPHA
-LOCK< NICHT VERGESSEN"
510 CALL KEY(0,K,S):: IF S=0
 THEN 510 :: IF K=49 THEN 14
80 :: IF K=50 THEN 520 ELSE
510
520 CALL CLEAR :: CALL CHARS
ET :: FOR I=2 TO 8 :: CALL C
OLOR(I,16,2):: NEXT I
530 FOR I=9 TO 11 :: CALL CO
LOR(I,3,2):: NEXT I
540 CALL CHAR(120,"3C7EFFE7E
7FF7EDB")
550 CALL CHAR(104,"995A3CFFF
F3C5A99",105,"81422418182442
81",106,"000000FFFF000000",1
07,"1818181818181818")
560 CALL CHAR(108,"191A3C3F3
F3C1A19",109,"98583CFCFC3C58
98",110,"00003CFFFF3C5A99",1
11,"995A3CFFFF3C0000")
570 CALL CHAR(112,"00003C3F3
F3C1A19",113,"00003CFCFC3C58
98",114,"191A3C3F3F3C0000",1
15,"98583CFCFC3C0000")
580 DISPLAY AT(2,6):"pjnjnjn
jnjnjnjnjnjq"
590 FOR I=3 TO 19 STEP 2 ::
DISPLAY AT(I,6):"kikikikikik
ikikikik" :: NEXT I
600 FOR I=4 TO 18 STEP 2 ::
DISPLAY AT(I,6):"ljhjhjhjhjh
jhjhjhjm" :: NEXT I
610 DISPLAY AT(20,6):"rjojoj
ojojojojojojs"
620 DISPLAY AT(24,1):" "
630 RANDOMIZE :: V=0 :: FE=0
 :: A=0 :: B=0 :: C=0 :: D=0
 :: E=0
640 Z1=INT(10*RND)*2+2 :: S1
=INT(10*RND)*2+8
650 Z2=INT(10*RND)*2+2 :: S2
=INT(10*RND)*2+8
660 Z3=INT(10*RND)*2+2 :: S3
=INT(10*RND)*2+8
670 Z4=INT(10*RND)*2+2 :: S4
=INT(10*RND)*2+8
680 E=Z1+S1+Z2+S2+Z3+S3+Z4+S
4
690 IF Z1=Z2 AND S1=S2 THEN
630 :: IF Z1=Z3 AND S1=S3 TH
EN 630 :: IF Z1=Z4 AND S1=S4
 THEN 630
700 IF Z2=Z3 AND S2=S3 THEN
630 :: IF Z2=Z4 AND S2=S4 TH
EN 630 :: IF Z3=Z4 AND S3=S4
 THEN 630
710 N=65 :: FOR I=2 TO 20 ST
EP 2 :: CALL VCHAR(I,7,N,1):
: N=N+1 :: NEXT I
720 DISPLAY AT(21,6):"1 2 3
4 5 6 7 8 9 10"
730 DISPLAY AT(23,8)SIZE(12)
:"VERSUCH NR.:" :: DISPLAY A
T(23,20):V :: CALL SOUND(400
,262,3,261,4,263,4)
740 CALL CHAR(33,"3C42A58181
A5423C"):: CALL SPRITE(#1,33
,16,9,57):: Z=2 :: SP=8 :: G
OTO 800
750 Z=2 :: GOTO 800
760 Z=20 :: GOTO 800
770 SP=8 :: GOTO 800
780 SP=26 :: GOTO 800
790 SP=8
800 CALL JOYST(1,X,Y):: X=SG
N(X):: Y=-SGN(Y)
810 SP=SP+2*X :: IF SP<8 THE
N 770 :: IF SP>26 THEN 780 :
: Z=Z+2*Y :: IF Z<2 THEN 750
 :: IF Z>20 THEN 760 :: CALL
 LOCATE(#1,Z*8-7,SP*8-7)
820 CALL KEY(1,K,S):: IF S=0
 THEN 800 :: IF K=18 THEN 83
0
830 CALL SOUND(200,262,0)
```

```
840 V=V+1 :: DISPLAY AT(23,2
0):V :: CALL POSITION(#1,X1,
Y1):: N=48
850 IF X1=Z1*8-7 AND Y1=S1*8
-7 THEN 1180
860 IF X1=Z2*8-7 AND Y1=S2*8
-7 THEN 1190
870 IF X1=Z3*8-7 AND Y1=S3*8
-7 THEN 1200
880 IF X1=Z4*8-7 AND Y1=S4*8
-7 THEN 1210
890 IF X1=Z1*8-7 AND Y1<>S1*
8-7 THEN GOSUB 1340 ELSE IF
X1<>Z1*8-7 AND Y1=S1*8-7 THE
N GOSUB 1340
900 FOR I=0 TO 20 STEP 2
910 IF X1=(Z1-I)*8-7 AND Y1=
(S1-I)*8-7 THEN GOSUB 1340
920 IF X1=(Z1-I)*8-7 AND Y1=
(S1+I)*8-7 THEN GOSUB 1340
930 IF X1=(Z1+I)*8-7 AND Y1=
(S1+I)*8-7 THEN GOSUB 1340
940 IF X1=(Z1+I)*8-7 AND Y1=
(S1-I)*8-7 THEN GOSUB 1340
950 NEXT I
960 IF X1=Z2*8-7 AND Y1<>S2*
8-7 THEN GOSUB 1340 ELSE IF
X1<>Z2*8-7 AND Y1=S2*8-7 THE
N GOSUB 1340

970 FOR I=0 TO 20 STEP 2
980 IF X1=(Z2-I)*8-7 AND Y1=
(S2-I)*8-7 THEN GOSUB 1340
990 IF X1=(Z2-I)*8-7 AND Y1=
(S2+I)*8-7 THEN GOSUB 1340
1000 IF X1=(Z2+I)*8-7 AND Y1
=(S2+I)*8-7 THEN GOSUB 1340
1010 IF X1=(Z2+I)*8-7 AND Y1
=(S2-I)*8-7 THEN GOSUB 1340
1020 NEXT I
1030 IF X1=Z3*8-7 AND Y1<>S3
*8-7 THEN GOSUB 1340 ELSE IF
 X1<>Z3*8-7 AND Y1=S3*8-7 TH
EN GOSUB 1340
1040 FOR I=0 TO 20 STEP 2
1050 IF X1=(Z3-I)*8-7 AND Y1
=(S3-I)*8-7 THEN GOSUB 1340
1060 IF X1=(Z3-I)*8-7 AND Y1
=(S3+I)*8-7 THEN GOSUB 1340
1070 IF X1=(Z3+I)*8-7 AND Y1
=(S3+I)*8-7 THEN GOSUB 1340
1080 IF X1=(Z3+I)*8-7 AND Y1
=(S3-I)*8-7 THEN GOSUB 1340
1090 NEXT I
1100 IF X1=Z4*8-7 AND Y1<>S4
*8-7 THEN GOSUB 1340 ELSE IF
 X1<>Z4*8-7 AND Y1=S4*8-7 TH
EN GOSUB 1340
1110 FOR I=0 TO 20 STEP 2
1120 IF X1=(Z4-I)*8-7 AND Y1
=(S4-I)*8-7 THEN GOSUB 1340
1130 IF X1=(Z4-I)*8-7 AND Y1
=(S4+I)*8-7 THEN GOSUB 1340
1140 IF X1=(Z4+I)*8-7 AND Y1
=(S4+I)*8-7 THEN GOSUB 1340
1150 IF X1=(Z4+I)*8-7 AND Y1
=(S4-I)*8-7 THEN GOSUB 1340
1160 NEXT I
1170 CALL HCHAR((X1+7)/8,(Y1
+7)/8,N,1):: GOTO 740
1180 A=Z1+S1 :: GOSUB 1350 :
: GOSUB 1470 :: GOTO 740
1190 B=Z2+S2 :: GOSUB 1380 :
: GOSUB 1470 :: GOTO 740
1200 C=Z3+S3 :: GOSUB 1410 :
: GOSUB 1470 :: GOTO 740
1210 D=Z4+S4 :: GOSUB 1440 :
: GOSUB 1470 :: GOTO 740
1220 CALL CLEAR :: CALL SOUN
D(100,1661,2):: CALL SOUND(1
00,1661,2):: CALL SOUND(100,
1480,2)
1230 CALL SOUND(100,1319,2):
: CALL SOUND(100,1319,2):: C
ALL SOUND(100,1175,2):: CALL
 SOUND(100,1108,2)
1240 CALL SOUND(100,1108,2):
: CALL SOUND(100,988,2):: CA
LL SOUND(100,988,2):: CALL S
OUND(300,880,0,879,1,881,1)
1250 CALL CLEAR :: DISPLAY A
T(8,2)SIZE(10):"DU HATTEST"
:: DISPLAY AT(8,13)SIZE(3):V
 :: DISPLAY AT(8,18):"VERSUC
HE."

1260 IF V>25 THEN 1270 :: IF
 V>20 THEN 1280 :: IF V>15 T
HEN 1290 :: IF V>10 THEN 130
0 :: IF V>5 THEN 1310
1270 DISPLAY AT(10,2):"DAS W
AR SEHR SCHWACH!" :: GOTO 13
20
1280 DISPLAY AT(10,2):"DU MU
SST NOCH VIEL UEBEN!" :: GOT
O 1320
1290 DISPLAY AT(10,2):"DAS W
AR SCHON GANZ GUT!" :: GOTO
1320
1300 DISPLAY AT(10,2):"DAS W
AR EXTRAKLASSE!" :: DISPLAY
AT(12,2):"FAST PERFEKT" :: G
OTO 1320
1310 DISPLAY AT(10,2):"DAS W
AR REINER ZUFALL!" :: DISPLA
Y AT(14,2):"TROTZDEM GRATULI
ERE ICH" :: DISPLAY AT(16,2)
:"ZUM OBERGALAXISCOMMANDER"
```

```
1320 DISPLAY AT(24,6):"NOCH
EIN SPIEL? J/N"
1330 CALL KEY(0,K,S):: IF S=
0 THEN 1330 :: IF K=106 THEN
 520 :: IF K=110 THEN 1680 E
LSE 1330
1340 N=N+1 :: RETURN
1350 FOR I=30 TO 0 STEP -1 :
: CALL SOUND(75,-1,I):: NEXT
 I
1360 CALL DELSPRITE(#1):: CA
LL COLOR(12,14,2):: CALL HCH
AR(Z1,S1,120,1)
1370 FOR I=0 TO 30 :: CALL S
OUND(100,523,I):: NEXT I ::
RETURN
1380 FOR I=30 TO 0 STEP -1 :
: CALL SOUND(75,-1,I):: NEXT
 I
1390 CALL DELSPRITE(#1):: CA
LL COLOR(12,14,2):: CALL HCH
AR(Z2,S2,120,1)
1400 FOR I=0 TO 30 :: CALL S
OUND(100,523,I):: NEXT I ::
RETURN
1410 FOR I=30 TO 0 STEP -1 :
: CALL SOUND(75,-1,I):: NEXT
 I
1420 CALL DELSPRITE(#1):: CA
LL COLOR(12,14,2):: CALL HCH
AR(Z3,S3,120,1)
1430 FOR I=0 TO 30 :: CALL S
OUND(100,523,I):: NEXT I ::
RETURN
1440 FOR I=30 TO 0 STEP -1 :
: CALL SOUND(75,-1,I):: NEXT
 I
1450 CALL DELSPRITE(#1):: CA
LL COLOR(12,14,2):: CALL HCH
AR(Z4,S4,120,1)
1460 FOR I=0 TO 30 :: CALL S
OUND(100,523,I):: NEXT I ::
RETURN
1470 FE=A+B+C+D :: IF FE=E T
HEN 1220 :: RETURN
1480 PRINT "ALS KOMMANDAND E
INER RAUM-" :: PRINT :: PRIN
T "PATROUILLE SOLLST DU IM U
NI-" :: PRINT
1490 PRINT "VERSUM 4 FEINDLI
CHE RAUM-" :: PRINT :: PRINT
 "SCHIFFE ENTDECKEN." :: PRI
NT
1500 PRINT "MIT DEM JOYSTICK
 KANNST DU" :: PRINT :: PRIN
T "DIE JEWEILIGEN KOORDINATE
N-" :: PRINT
1510 PRINT "PUNKTE ANFLIEGEN
. MIT DER" :: PRINT :: PRINT
 "FEUERTASTE AKTIVIERST DU"
:: PRINT
1520 PRINT "DEIN SICHTGERAET
." :: PRINT :: PRINT "------
----------------------" :: P
RINT "WEITER MIT TASTE >SPAC
E<" :: PRINT
1530 CALL KEY(0,K,S):: IF S=
0 THEN 1530 :: CALL CLEAR
1540 PRINT "NACH KURZER ZEIT
 WIRD DIR" :: PRINT :: PRINT
 "ANGEZEIGT, WIEVIEL FEINDE
DU" :: PRINT
1550 PRINT "VON DEINEM STAND
PUNKT AUS" :: PRINT :: PRINT
 "SEHEN KANNST." :: PRINT
1560 PRINT "DIE ANZEIGE ZAEH
LT HIERBEI" :: PRINT :: PRIN
T "ALLE FEINDE SOWOHL IN DIA
GO-" :: PRINT
1570 PRINT "NALER, HORIZONTA
LER ALS AUCH" :: PRINT :: PR
INT "IN VERTIKALER RICHTUNG.
" :: PRINT
1580 PRINT "----------------
------------" :: PRINT "WEIT
ER MIT TASTE >SPACE<" :: PRI
NT :: ::
1590 CALL KEY(0,K,S):: IF S=
0 THEN 1590 :: CALL CLEAR
1600 PRINT "EIN FEIND WIRD N
UR SICHTBAR," :: PRINT :: PR
INT "WENN DU UEBER IHM STEHS
T UND" :: PRINT
1610 PRINT "DIE FEUERTASTE D
RUECKST." :: PRINT
1620 PRINT "DURCH UEBERLEGUN
G KANNST DU" :: PRINT :: PRI
NT "DIE ANZAHL DEINER VERSUC
HE" :: PRINT :: PRINT "ERHEB
LICH REDUZIEREN." :: PRINT
1630 PRINT "NUR MIT EINEM GU
TEN ERGEBNIS" :: PRINT :: PR
INT "KANNST DU STELLVERTRETE
R VON" :: PRINT
1640 PRINT "PERRY RHODAN WER
DEN." :: PRINT
1650 PRINT :: PRINT "VIEL GL
UECK COMMANDER" :: PRINT ::
::
1660 PRINT "----------------
------------DRUECKE >SPACE<
FUER START"
1670 CALL KEY(0,K,S):: IF S=
0 THEN 1670 :: IF K=32 THEN
520 ELSE 1670
1680 CALL CLEAR :: END
```

Spiel ohne Grenzen

Ich möchte Ihnen ein in GPL geschriebenes Modul vorstellen, das in außergewöhnlichem Maß mit den Power-Up-Routinen des TI herumspielt und ihnen einige Überraschungen entlockt.
Der eigentliche Nutzen ist daher relativ gering, es handelt sich um ein reines Demo-Modul. Es spielt Musikstücke ab, wahlweise Beethovens Neunte oder Alohaoe.
Egal, ob Sie im Titelbild, Hauptmenü oder TI-BASIC sind, ob ein Programm läuft oder Sie gerade programmieren: Solange das Modul steckt, werden Ihre Nerven nicht geschont.
Nur im Hauptmenü haben Sie die Möglichkeit, den Punkt „Angenehme Ruhe" zu wählen. Wenn Sie etwas wählen, was bereits gewählt ist, weist der TI mit kurzen Textzeilen, die er ins Hauptmenü einblendet, darauf hin. Auch das ist ein Verfahren, das bisher kaum jemand kennt.
Achtung: Die Syntax des abgedruckten GPL-Listings entspricht nicht derjenigen, die Heiner Martin in seinem ROM-Listing, und die ich in den GPL-Einführungen bisher benutzt habe. Die Syntax entspricht statt dessen den Anforderungen des GPL-Assemblers von Michael Weiand. Es ist ohne Veränderungen assemblierbar.
Der Objektcode kann entweder in ein EPROM gebrannt oder aber ins PROGRAM-Format für ein Supermodul umgesetzt werden. In jedem Fall aber muß im ROM-Bereich mindestens von >7500 bis >7700 ein RAM vorhanden sein. Bei Verwendung eines EPROMs und Heiner Martins Modulplatine ist der freie ROM-Steckplatz also mit einem RAM zu besetzen, sonst gibt's nur Systemabstürze.
Zur Funktion des Moduls: Im Modulheader ist das MSB des zweiten Byte gesetzt. Das bedeutet, der TI ruft unmittelbar nach dem Power-Up und noch vor der Abfrage der INIT-Routinen der einzelnen Peripheriegeräte das GROM unter G6010 auf (siehe ROM-Listing G0379 bis G0392). Die dort stehende Routine benutzt die Speicherstelle >7600 als Flag. Das zeigt, welches Musikstück gespielt werden soll.
>0000 steht für Beethovens Neunte, >0002 für Alohaoe und >0004 für gar kein Stück. Jeder andere Wert, wie er direkt nach dem Einsetzen des Moduls auftreten kann, wird auf >0000 geändert. Die Routine verändert das TI-Zeichen, schreibt zusätzliche Texte ins Titelbild, lädt die Interrupt-Routine ab >7500 ins RAM und startet sie durch Setzen des User-Defined-Interrupt in >83C4.
Diese Interrupt-Routine sieht jedesmal nach, ob >83CE noch ungleich Null ist, ob also die Soundverarbeitung noch andauert. Wenn nicht, legt sie durch Herunterzählen der Speicherstelle >7604 eine kurze Pause ein, lädt dann >83CC und >83CE mit neuen Werten (>83CE enthält die Startadresse der gewünschten Soundliste) und setzt den Zähler >7604 neu (>7606 enthält die Länge der Pause in 1/50 sec).
Die pure Wiederholung eines Stükkes ließe sich zwar auch ohne die Interrupt-Routine realisieren. So hat es aber den Vorteil, daß auch nach erzwungenen Unterbrechungen, etwa durch den Beep beim Tastendruck im Titelbild und Hauptmenü oder den Fehlerton im TI-BASIC, die Soundverarbeitung fortgesetzt wird.
Wählen Sie im Hauptmenü einen Punkt des Moduls an, so passiert nichts weiter, als daß die Pointer >7600 und >7602 neu gesetzt werden und ein Reset ausgeführt wird. Es sei denn, Sie wählen etwas, das bereits gewählt ist. Dann schreibt das Modul einen kurzen Hinweis ins Hauptmenü und ruft dieses neu auf. Im ROM-Listing wäre die Einsprungstelle dafür G01B6.
Da es aber verschiedene Versionen des GROMs 0 gibt, würde ein B >01B6 bei vielen Rechnern zum Absturz führen. Bei meinem Rechner zum Beispiel ist das gesamte Listing um drei Byte verschoben, so daß bei mir die Einsprungstelle G01B9 ist. Anderen geht es genauso. Deshalb sucht sich das Modul die Einsprungstelle, die direkt nach der Anweisung ALL >20 liegt, jedesmal selbst. Ab G01A0 sucht es das GROM 0 nach dem Byte >20 ab und springt an die Stelle danach ein. Das GROM 0 baut das Hauptmenü dann erneut auf und läßt Sie wieder wählen. *Sven Dyroff*□

```
       GROM >6000
*----------------------------------
       DATA >AAFF,>0000,>0000,NAMES
       DATA >0000,>0000,>0000,>0000
*----------------------------------
       DCEQ  >0004,§>7600                   * Ruhe ?
       BS    STILLE                         * Ja: Sprung
       MOVE  >0048,GROM§ZEICHEN,VDP§>0808   * TI-Symbol umdefinieren
       MOVE  >0014,GROM§TITEL1,VDP§>02C8    * 1. Zeile neu schreiben
       MOVE  >0012,GROM§TITEL2,VDP§>02EB    * 2. Zeile neu schreiben
       DCZ   §>7600                         * Beethovens Neunte ?
       BS    CONT1                          * Ja: Sprung
       DCEQ  >0002,§>7600                   * Alohaoe ?
       BS    CONT1                          * Ja: Sprung
       DCLR  §>7600                         * Sonstiges: Spiele Beethovens 9.
CONT1  MOVE  >0040,GROM§INTER,§>7500        * Interruptroutine laden
CONT2  DST   SOUND1,§>7602                  * Startadresse der Soundliste B. 9.
       DCZ   §>7600                         *
       BS    CONT4
       DST   SOUND2+1,§>7602                * Startadresse der Soundliste Aloha
CONT4  DST   >0001,§>7604                   * Zähler setzen
       DST   >0064,§>7606                   * Pausenlänge setzen 2 sec.
       DST   >7500,§>83C4                   * User-Defined-Interrupt setzen
       RTN                                  * Ende
*----------------------------------
NAMES  DATA  NAME2,RUHE                     * Liste für Hauptmenü
       BYTE  >0E
       TEXT  'ANGENEHME RUHE'
NAME2  DATA  NAME3,ALOHA
       BYTE  >07
       TEXT  'ALOHAOE'
NAME3  DATA  >0000,NEUNTE
       BYTE  >11
       TEXT  'BEETHOVENS NEUNTE'
*----------------------------------
NEUNTE DCZ   §>7600                         * Wird B. 9. bereits gespielt ?
       BS    SCHON                          * Ja: Sprung
       DCLR  §>7600                         * Nein: Spiele sie!
       BR    CONT3
*----------------------------------
ALOHA  DCEQ  >0002,§>7600                   * Wird Alohaoe bereits gespielt ?
       BS    SCHON                          * Ja: Sprung
       DST   >0002,§>7600                   * Nein: Spiele es!
       BR    CONT3
*----------------------------------
RUHE   DCEQ  >0004,§>7600                   * Ist bereits Ruhe ?
       BS    RUHIG                          * Ja: Sprung
       DST   >0004,§>7600                   * Nein: Klappe halten!
*----------------------------------
CONT3  INCT  §>8373                         * Der RTN am Ende von CONT2 muß als
       DST   >0020,*>8373                   * EXIT ausgeführt werden, deshalb
       BR    CONT2                          * die GROM-Reset-Adr. auf Substack
*----------------------------------
* Das spiele ich doch gerade!
*----------------------------------
SCHON  MOVE  >001B,GROM§HIRNI1,VDP§>02A2    * Text schreiben
       MOVE  >000B,GROM§HIRNI2,VDP§>02E2
       B     WEITER
*----------------------------------
* Ich bin ja schon ruhig!
```

```
*-----------------------------------
RUHIG   MOVE   >0017,GROM§PRRUHE,VDP§>02C2      * Text schreiben
        DCLR   §>83C4                           * User-Defined-Interrupt löschen
*-----------------------------------
* Einsprung für Neuaufbau des
* Hauptmenüs suchen und ausführen
*-----------------------------------
WEITER DST     >01A0,§>8300                     * Ab >01A0 wird gesucht
WEITER2 MOVE    >0001,GROM§>0000(§>8300),§>8302 *Ein Byte holen
        DINC   §>8300                           * Adresse erhöhen
        CEQ    >20,§>8302                       * Ist es der Operand von ALL >20 ?
        BR     WEITER2                          * Nein: weiter suchen
        INCT   §>8373                           * Ja: Adresse auf den Substack
        DST    §>8300,*>8373
        RTN                                     * Und anspringen
*-----------------------------------
* Option 3:  Angenehme Ruhe
*-----------------------------------
STILLE MOVE    >000F,GROM§BANAUSE,VDP§>0169     * Text schreiben
        RTN
*-----------------------------------
* Chars für die Geige
*-----------------------------------
ZEICHEN DATA   >0000,>000E,>191C,>0300
        DATA   >0000,>0000,>8060,>19C6
        DATA   >0C12,>2945,>A294,>48B0
        DATA   >0000,>0102,>040B,>3444
        DATA   >350E,>D72A,>1428,>4484
        DATA   >C0E0,>18C4,>320E,>0000
        DATA   >A5BA,>859A,>8A63,>201F
        DATA   >2428,>D020,>4040,>8000
        DATA   >0000,>0000,>0000,>0000
*-----------------------------------
* Texte für Titelbild und Hauptmenü
*-----------------------------------
TITEL1 TEXT    '6  BY SOUNDLOVER     '
TITEL2 TEXT    'TEL.: 06121/462686'
BANAUSE TEXT    '*KUNSTBANAUSE!*'
HIRNI1 TEXT    'DAS SPIELE ICH DOCH GERADE,'
HIRNI2 TEXT    'SIE HIRNIE!'
PRRUHE TEXT    'ICH BIN JA SCHON RUHIG!'
*-----------------------------------
* Maschinenroutine für Interrupt-
* steuerung
*-----------------------------------
INTER  DATA    >D020,>83CE           * MOVB §>83CE,R0
        DATA   >160E                 * JNE  $+30
        DATA   >0620,>7604           * DEC  §>7604
        DATA   >160B                 * JNE  $+24
        DATA   >C820,>7602,>83CC     * MOV  §>7602,§>83CC
        DATA   >D820,>7524,>83CE     * MOV  §>7524,§>83CE
        DATA   >C820,>7606,>7604     * MOV  §>7606,§>7604
        DATA   >024E,>FFFE           * ANDI R14,>FFEE
        DATA   >045B                 * B    *R11
        DATA   >0100                 * DATA >0100
*-----------------------------------
* Soundliste  Beethovens Neunte
*-----------------------------------
SOUND1 DATA >0989,>0A9E,>A50D,>BDCA
        DATA >1ADD,>1E03,>9FBF,>DF01
```

```
DATA >059E,>BDC2,>15DD,>1E03
DATA >9FBF,>DF01,>0980,>0A9E
DATA >AE0B,>BDCE,>0FDD,>1E03
DATA >9FBF,>DF01,>098E,>089E
DATA >A90A,>BDCD,>11DD,>1E03
DATA >9FBF,>DF01,>059E,>BDCA
DATA >1ADD,>1E03,>9FBF,>DF01
DATA >0980,>0A9E,>AE0B,>BDCC
DATA >1FDD,>1E03,>9FBF,>DF01
DATA >0989,>0A9E,>A50D,>BDCA
DATA >23DD,>1E03,>9FBF,>DF01
DATA >098E,>0B9E,>A20E,>BDC4
DATA >1CDD,>1E03,>9FBF,>DF01
DATA >0985,>0D9E,>AE0F,>BDC2
DATA >15DD,>1E03,>9FBF,>DF01
DATA >059E,>BDCA,>1ADD,>1E03
DATA >9FBF,>DF01,>098E,>0B9E
DATA >A20E,>BDCF,>27DD,>1E03
DATA >9FBF,>DF01,>0989,>0A9E
DATA >A50D,>BDCC,>1FDD,>1E03
DATA >9FBF,>DF01,>039E,>BDDD
DATA >2D03,>9FBF,>DF01,>098E
DATA >0B9E,>A20E,>BDCA,>23DD
DATA >0E03,>9FBF,>DF01,>039E
DATA >BDDD,>3C03,>9FBF,>DF01
DATA >0989,>0A9E,>A50D,>BDCA
DATA.>1ADD,>1E03,>9FBF,>DF01
DATA >059E,>BDC2,>15DD,>1E03
DATA >9FBF,>DF01,>0980,>0A9E
DATA >AE0B,>BDCE,>0FDD,>1E03
DATA >9FBF,>DF01,>098E,>089E
DATA >A90A,>BDCD,>11DD,>1E03
DATA >9FBF,>DF01,>059E,>BDCA
DATA >1ADD,>1E03,>9FBF,>DF01
DATA >0980,>0A9E,>AE0B,>BDCC
DATA >1FDD,>1E03,>9FBF,>DF01
DATA >0989,>0A9E,>A50D,>BDCA
DATA >23DD,>1E03,>9FBF,>DF01
DATA >098E,>0B9E,>A20E,>BDC4
DATA >1CDD,>1E03,>9FBF,>DF01
DATA >0985,>0D9E,>AE0F,>BDC2
DATA >15DD,>1E03,>9FBF,>DF01
DATA >059E,>BDCA,>1ADD,>1E03
DATA >9FBF,>DF01,>098E,>0B9E
DATA >A70E,>BDCF,>27DD,>1E03
DATA >9FBF,>DF01,>0989,>0A9E
DATA >A50D,>BDCC,>1FDD,>1E03
DATA >9FBF,>DF01,>098E,>0B9E
DATA >A70E,>BDCA,>23DD,>2D03
DATA >9FBF,>DF01,>0985,>0D9E
DATA >AD11,>BDC5,>35DD,>0E03
DATA >9FBF,>DF01,>039E,>BDDD
DATA >3C03,>9FBF,>DF01,>068E
DATA >0B9E,>A20E,>BD1E,>029F
DATA >BF01,>029E,>BD1E,>029F
DATA >BF01,>0689,>0A9E,>A50D
DATA >BD1E,>029F,>BF01,>0685
DATA >0D9E,>AE0F,>BD1E,>029F
DATA >BF01,>068E,>0B9E,>A20E
DATA >BD1E,>029F,>BF01,>0689
DATA >0A9E,>A50D,>BD0E,>029F
DATA >BF01,>0681,>0A9E,>AE0B
DATA >BD0E,>029F,>BF01,>0689
DATA >0A9E,>A50D,>BD1E,>029F
DATA >BF01,>0685,>0D9E,>AE0F
DATA >BD1E,>029F,>BF01,>068E
DATA >0B9E,>A20E,>BD1E,>029F
DATA >BF01,>0689,>0A9E,>A50D
DATA >BD0E,>029F,>BF01,>0680
DATA >0A9E,>AE0B,>BD0E,>029F
DATA >BF01,>0689,>0A9E,>A50D
DATA >BD1E,>029F,>BF01,>068E
DATA >0B9E,>A20E,>BD1E,>029F
DATA >BF01,>0685,>0D9E,>AE0F
DATA >BD1E,>029F,>BF01,>068E
DATA >0B9E,>A20E,>BD1E,>029F
DATA >BF01,>048D,>119E,>BF3C
DATA >019F,>0109,>890A,>9EA5
DATA >0DBD,>CA1A,>DD1E,>039F
DATA >BFDF,>0105,>9EBD,>C215
DATA >DD1E,>039F,>BFDF,>0109
DATA >800A,>9EAE,>0BBD,>CE0F
DATA >DD1E,>039F,>BFDF,>0109
DATA >8E08,>9EA9,>0ABD,>CD11
DATA >DD1E,>039F,>BFDF,>0105
DATA >9EBD,>CA1A,>DD1E,>039F
DATA >BFDF,>0109,>800A,>9EAE
DATA >0BBD,>CC1F,>DD1E,>039F
DATA >BFDF,>0109,>890A,>9EA5
DATA >0DBD,>CA23,>DD1E,>039F
DATA >BFDF,>0109,>8E0B,>9EA2
DATA >0EBD,>C41C,>DD1E,>039F
DATA >BFDF,>0109,>850D,>9EAE
DATA >0FBD,>C215,>DD1E,>039F
DATA >BFDF,>0105,>9EBD,>CA1A
DATA >DD1E,>039F,>BFDF,>0109
DATA >8E0B,>9EA2,>0EBD,>CF27
DATA >DD1E,>039F,>BFDF,>0109
DATA >890A,>9EA5,>0DBD,>CC1F
DATA >DD1E,>039F,>BFDF,>0107
DATA >8E0B,>9EBD,>CA23,>DD2D
DATA >039F,>BFDF,>0109,>850D
DATA >9EAD,>11BD,>C535,>DD0E
DATA >039F,>BFDF,>0103,>9EBD
DATA >DD3C,>039F,>BFDF,>0000
*---------------------------------
* Soundliste  Alohaoe
*---------------------------------
SOUND2 DATA >0003,>9FBF,>DF01
       DATA >098D,>119C,>A300,>B9C3
       DATA >00D9,>1203,>9FBF,>DF01
       DATA >0985,>0D9C,>A300,>B9C3
       DATA >00D9,>1203,>9FBF,>DF01
       DATA >0989,>0A9C,>AA23,>B9C3
       DATA >00D9,>1203,>9FBF,>DF01
       DATA >0989,>0A9C,>AA1A,>B9C2
```

```
DATA >15D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0C9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C2
```

```
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>A300,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0980,>0A9C,>AC1F,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>AA1A,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098D,>119C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>AA1A,>B9C2
DATA >15D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >098E,>0F9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >0982,>0E9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0985,>0D9C,>A41C,>B9C0
DATA >14D9,>1203,>9FBF,>DF01
DATA >098E,>0B9C,>AA23,>B9C3
DATA >00D9,>1203,>9FBF,>DF01
DATA >0989,>0A9C,>A41C,>B9C0
```

Fortsetzung von Seite 30

```
        DATA >14D9,>1203,>9FBF,>DF01
        DATA >098E,>0B9C,>AA23,>B9C3
        DATA >00D9,>1203,>9FBF,>DF01
        DATA >0982,>0E9C,>A41C,>B9C0
        DATA >14D9,>1203,>9FBF,>DF01
        DATA >0985,>0D9C,>AA23,>B9C3
        DATA >00D9,>1203,>9FBF,>DF01
        DATA >0985,>0D9C,>AA1A,>B9C2
        DATA >15D9,>1203,>9FBF,>DF01
        DATA >0985,>0D9C,>AA23,>B9C3
        DATA >00D9,>1203,>9FBF,>DF01
        DATA >0985,>0D9C,>AA1A,>B9C2
        DATA >15D9,>1203,>9FBF,>DF01
        DATA >0985,>0D9C,>AA23,>B9C3
        DATA >00D9,>1203,>9FBF,>DF01
        DATA >0985,>0D9C,>AA1A,>B9C2
        DATA >15D9,>1203,>9FBF,>DF00
        DATA >0000
*--------------------------------
        END
```

DEM TI UNTERS HEMD GESCHAUT, TEIL IV

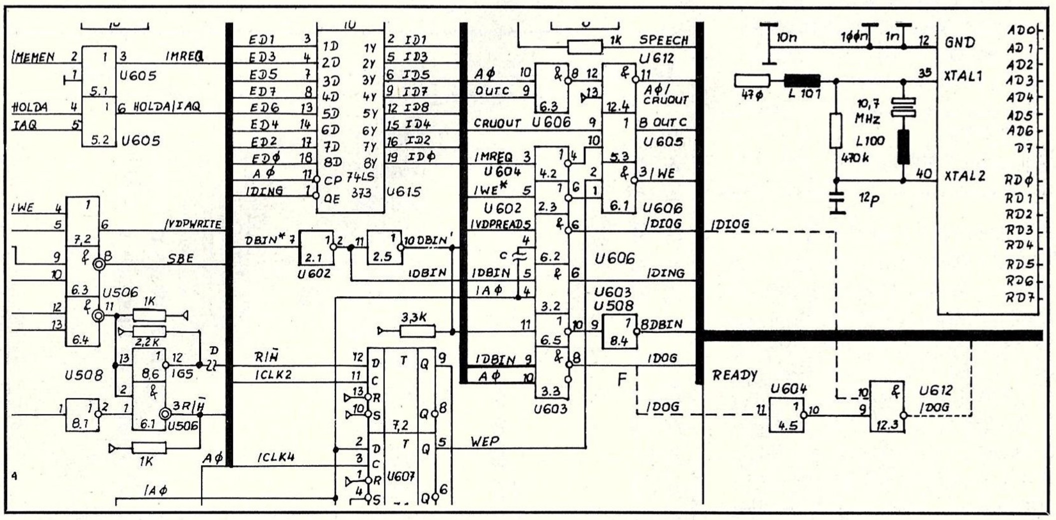
Bus-Fahrt

2.3 Beschleunigung der über den externen Datenbus laufenden allgemeinen Acht-Bit-I/O-Transfers

Diese Betriebsart wird dann interessant, wenn man sich von den langsamen GROMs trennt und statt dessen die universelle GROM-GRAM-Steckeinheit verwendet, die im Abschnitt 3, Revolution, ausführlich dokumentiert wird. Dort werden Speicher mit erheblich kürzeren Zugriffszeiten eingesetzt, so daß die Synchronisation des Zugriffs nicht mehr über die Ablaufsteuerung vorgenommen werden muß. Es ist ein Zugriff innerhalb von zwei T-Zyklen (bei langsamen Eproms drei T-Zyklen) möglich, wie der im Abschnitt Anotomie beschriebene interne 16 Bit breite Zugriff, nur mit dem Unterschied, daß er nicht intern und nur acht Bit breit auf dem Highteil des Datenbusses der CPU laufen muß. Das kann erreicht werden, wenn man solche Zugriffe auch mit in die Zusatzlogik aus Punkt 2.2 einspeist, so daß auch dann die Ablaufsteuerung am Start gehindert wird, zusätzlich muß aber noch für die Selektierung des bidirektionalen Treibers U 614, 74LS245 gesorgt werden, da jetzt die Quelle oder die Senke nicht in der Konsole, sondern am externen Datenbus liegt. Der schnelle Zugriff wird dann erreicht, wenn das externe Gerät das low-aktive Signal /EXIO liefert, das über den indirekten Steckverbinder in die Konsole gelangt. Das Signal /EXIO kann von mehreren externen Quellen geliefert werden und die diesen Eingang ansteuernden Ausgänge müssen demzufolge Open-Kollektor-Ausgänge oder Three-State-Ausgänge sein! Auf der Zusatzplatine wird die Leitung über einen Pull-Up-Widerstand nach +U5 gezogen, so daß im Normalfall dieser Eingang unwirksam ist. Nur wenn an diesem Eingang Low-Potential liegt, wird auf den schnellen Zugriff umgeschaltet.
Zusätzlich muß in diesem Fall dafür gesorgt werden,

SCHNELLE DATEN

daß die Daten auf den externen Bus kommen bzw. von dort gelesen werden können, das heißt, der bidirektionale Treiber U 614, 74LS245 muß selektiert werden. Das geschieht durch Auftrennen der Leitung /A0 vom Ausgang der IS U613.4 Pin 13 zum Gatter U606.2 Pin 4 am *Punkt C in Abbildung 1.* Diese Leitung wird mit der Bezeichnung CE zur Zusatzplatine geführt und dort an einem OR-Gatter B4 mit dem invertierten Signal /(/EXIO) verknüpft,



so daß der Ausgang CA nicht nur während der Phase von /A0 = High, sondern auch bei /EXIO = Low aktiviert wird:

/DIOG = /(/VDPREAK & (/A0 v EXIO))

Über das Gatter B3 wird im Falle eines Zugriffs mit aktivem Signal /EXIO die Ablaufsteuerung gestoppt.
Zur Vermeidung von Konflikten mit dem Treiber U 616 (74LS244) während eines Schreib-

ANMERKUNG DER REDAKTION

Wie Sie sicher bemerkt haben, sind in der vorletzten Ausgabe (HCA 4/88) die Schaltpläne und die beiden Layouts für die Platine zu diesem Artikel ziemlich „in die Hose" gegangen. Dies lag unter anderem daran, daß uns leider nicht die Originalzeichnungen, sondern nur ein Film vorgelegen hat. Aufgrund des Formates der Originale mußte der Plan in der HCA auch verkleinert wiedergegeben werden, was die Übersichtlichkeit nicht gerade verbessert.
Wir werden versuchen, bessere und größere Kopien der Pläne zu erstellen und sie den interessierten Lesern dann gegen Einsendung eines frankierten Rückumschlags zur Verfügung stellen.

zyklus' wird dieser am besten bei jeder Selektion des Treibers 74LS 245 abgeschaltet. Dazu wird das Selektionssignal &DOG von den Pins 1 und 19 der IS U 616 *(Punkt F in Abbildung 1)* abgetrennt, das Signal mit dem freien Inverter der IS U 604.3 negiert und an dem freien NAND-Gatter der IS U 612.3 mit dem Signal /DIOG verknüpft, der Ausgang des NAND-Gatters liefert dann das /CS-Signal für den 74LS244 genau dann:

/CS(74LS244)
 = /(DOG & /DIOG)

Die Verknüpfung wurde auf der Konsole vorgenommen, da sie auch ohne den Einbau der Zusatzplatine sinnvoll ist und dadurch ein weiterer Schaltkreis auf der Zusatzplatine eingespart werden konnte. Diese Änderung ist daher in Abbildung 1 auch schon enthalten.

LETZTE ERWEITERUNG

Als letzte Erweiterung wurde die Möglichkeit vorgesehen, durch Low-Ziehen des Eingangs GROMEN (GROM-Enable) die Selektion der GROMs am Modul-Schacht und in der Konsole abzuschalten. Das wird in dem Moment interessant, wenn mit dem Einsatz der universellen GROM-GRAM-Karte (Abschnitt 3, Revolution) auf die in der Konsole eingesetzten GROMs verzichtet wird (Ersatz durch Eproms und damit wesentlich schnellerer Zugriff), oder die vom Betriebssystem unterstützten 16 Ebenen GROM/GRAM genutzt werden sollen, die GROMs aber in ihren Fassungen verbleiben sollen, so daß nach Trennen der Konsole von der Erweiterungsbox wieder die internen GROMs angesprochen werden können. Das mit Inverter-Gatter D1 negierte Signal GROMEN wird am OR-Gatter B1 mit dem Signal /GS verknüpft. Dieses Signal wird am Ausgang der IS U508.6 Pin 12 gewonnen, der Ausgang wird zuvor von der Leitung /GS getrennt *Punkt D in Abbildung 1)* und als Signal DE zur Zusatzplatine geführt, das Ausgangssignal des OR-Gatters B1 wird als Signal DA auf die Leitung /GS der Konsole gelegt. Dadurch werden die GROMs genau dann selektiert, wenn:

/GS(GROM)
 = /(GS(IS508.6)
 & GROMEN)

Alle drei diskutierten Erweiterungen sind also in relativ einfacher Weise realisierbar und bringen doch eine erhebliche Erhöhung des Gebrauchswertes des Computers, neben der Erweiterung des nutzbaren Befehlssatzes der CPU eine erhebliche Steigerung der Arbeitsgeschwindigkeit des Rechners. Effektiv laufen Maschinenprogramme cirka doppelt so schnell auf der 16-Bit-Speichererweiterung wie auf der TI-Speichererweiterung. Die Buszyklen sind zwar dreimal so schnell, jedoch besteht die Ausführung eines Maschinenbefehls nicht nur aus Buszyklen, sondern auch aus „internen Zyklen" ohne Busverkehr. Unter diesem Aspekt sind die mechanischen Schwierigkeiten, die für den Umbau zu lösen sind, akzeptabel. Für die gesamte Umrüstung nach oben genannten Kriterien werden benötigt:

1 74LS00, 1 74LS02, 1 74LS32, 1 74LS138, 4 CMOS-RAM HM6264, 1 '4093, 1 '081, 1 npn-Transistor, diverse passive Bauelemente.

2.4 Belegung des indirekten I/O-Port-Steckverbinders am Beispiel eines 58poligen zweireihigen Steckverbinders

Der indirekte Steckverbinder wird direkt auf die Fahnen des direkten Steckverbinders aufgelötet. Der zweireihige Steckverbinder (Reihe A1...A29 und B1...B29) schließt hinten (Kontakte AB29) mit dem direkten Steckverbinder (Kontakte 43/44) ab, setzt vorne (Kontakte AB1) um sieben Stellen gegenüber dem direkten Steckverbinder (Kontakte 1/2) über.
Die übersetzenden Steckverbinderkontakte werden mit dem Konsolen-Print direkt verdrahtet.

Heinrich Cronjaeger □

Kontakt	TI	Funktion	Kontakt	TI	Funktion	
A 1	*	Masse	B 1	*	/HOLD	
A 2	*	Masse	B 2	*	/IS2 –	IDLE
A 3	*	Masse	B 3	*	/IS3 –	RESET
A 4	*	+UPUF	B 4	*	/IS4 –	n.b.
A 5	*	BE, U507.Pin 8	B 5	*	/IS5 –	XON
A 6	*	GROMEN	B 6	*	/IS7 –	LREX
A 7	*	/EXIO	B 7	*	/IS5 –	XOFF
A 8	2	SBE	B 8	1	+U5	
A 9	4	/EXT INT	B 9	3	/RESET	
A10	6	AB 5	B10	5	AB10	
A11	8	AB 4	B11	7	AB11	
A12	10	AB 12	B12	9	DBIN	
A13	12	READY-(/HOLD)	B13	11	AB 3	
A14	14	AB 7	B14	13	/LOAD	
A15	16	AB 1	B15	15	AB 2	
A16	18	AB 6	B16	17	AB 8	
A17	20	AB 13	B17	19	AB 0 / CRU-OUT	
A18	22	/CRUCLK	B18	21	Masse	
A19	24	/CLK3	B19	23	Masse	
A20	26	/WE (/WR)	B20	25	Masse	
A21	28	/MBE	B21	27	Masse	
A22	30	AB 14	B22	29	AB 9	
A23	32	/MEMEN (/REQ)	B23	31	AB 15	
A24	34	DB 0	B24	33	CRU-IN	
A25	36	DB 1	B25	35	DB 3	
A26	38	DB 2	B26	37	DB 7	
A27	40	DB 6	B27	29	DB 5	
A28	42	DB 4	B28	41	HOLDA/IAQ	
A29	44	SPEECH	B29	43	–U5	

DISKETTENLAUFWERKE

All Systems go

In der vorletzten Ausgabe waren wir beim Bus stehengeblieben. Und genau damit machen wir nun hier weiter.

Der Bus wurde mechanisch durch beidseitig an den Platinenrand geführte Leiterbahnen mit einem Abstand von 1/10 Zoll (=2.54 mm) realisiert. Auf diesen „Stecker" paßt dann ein weiblicher Platinensteckverbinder (2x17=34polig). Diese Steckverbinder sind in der Regel in der sogenannten Schneid-/Klemmtechnik ausgeführt, das heißt, man legt das dazugehörende 34polige Flachbandkabel (AWG 28/34pol.) in diese ein und preßt sie zusammen (Spezialzange oder Parallelschraubstock). Aus etwas Kabel und zwei dieser Stecker entsteht so ein professionelles Buskabel, welches oftmals überteuert erstanden werden muß. Die Pole des Busses werden 1:1 durchverbunden. Alle Laufwerke hängen parallel an diesem Bus. Für Drive 2, 3 und gegebenenfalls 4 ist daher nur ein weiterer Stecker aufzupressen. 3.5-Zoll-Laufwerke haben vielfach einen sogenannten Pfostensteckverbinder. Die Belegung ist identisch. Hier stehen auf der Driveplatine 34 Stifte ab, auf die der Pfostensteckverbinder (34polig), der seinerseits auf das Flachkabel aufgepreßt wird, gesteckt wird. Es handelt sich also nur um eine mechanische Abweichung, die aber besser ist als das Original, elektrisch aber vollkommen gleichwertig beschaltet ist. Man darf dem Pfostensteckverbindersystem aber, insbesondere nach häufigem Stecken, geringeren Verschleiß und damit niedrigere Übergangswiderstände nachsagen. Die Pole des Busses sind verbindlich mit Pin-Nummer 1 bis 34 bezeichnet, wobei, bei Draufsicht auf den Platinenstecker des Laufwerks von hinten, Pin 1 links unten, Pin 2 links oben, Pin 34 rechts oben und Pin 33 rechts unten angesiedelt sind. Die an den Kabelsteckern vorhandenen Kerben sollten, um Verwechslungen zu vermeiden, beim Montieren alle auf den gleichen

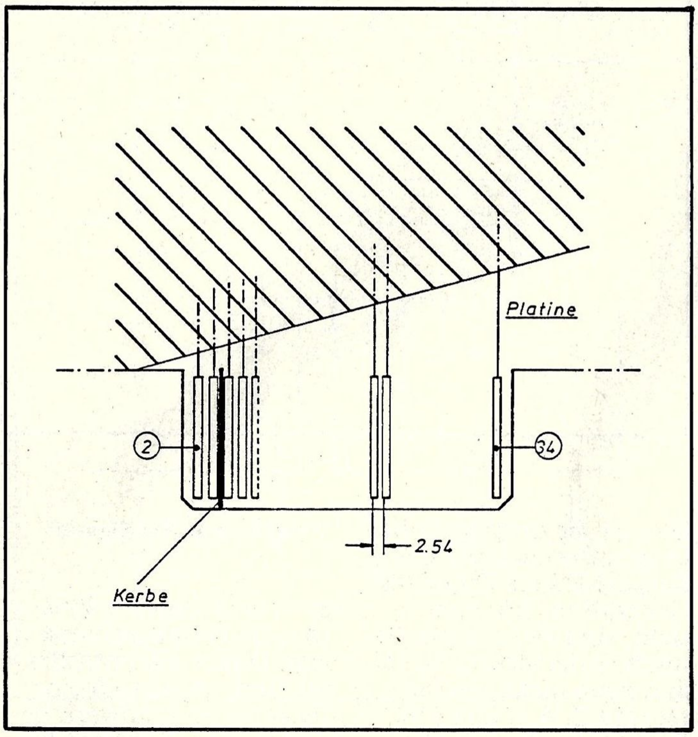


Abbildung 3: Busmechanik

(Abbildung in unterschiedlichen Maßstäben vergrößert; Bemaßung mit Originalmaßen.)

Pin (am besten Pin 1) zeigen. Sollte man sich dennoch einmal beim Aufstecken vertun (die Stekker passen auch, wenn man sie um 180 Grad dreht), kann nichts zerstört werden, es funktioniert nur nicht so, wie man es erwartet. Dies ergibt sich aus der elektrischen Realisation. Doch davon bald mehr. *Abbildung 3* zeigt die erwähnten *Busmechaniken*, die Tabelle in *Abbildung 4* die *elektrische Beschaltung* des Busses. Ähnlich wie bei den Daten und Handshakeleitungen der Centronics-Schnittstelle, hat jedes Bussignal seine eigene Masseleitung. Man darf sich aber nicht verrückt machen lassen. Alle Masseleitungen sind auf der Kontrollerplatine wie auf den Laufwerksplatinen miteinander verbunden. Wozu dann der Aufwand? Nun, bei der Verwendung der Flachbandkabel wechseln sich Signal- und Masseleitungen durch die Beschaltung der Pole ab. Dies verringert das sogenannte Signalübersprechen bei schnellen Signalen drastisch. Der Bus wird damit wasserdicht. Dies stellt sicher, daß die eingespeisten Signale nicht auf anderen Signalleitungen anzutreffen sind. Ein variables Disk-Operating-System (DOS) wird von der Hardware her erst dadurch das, was es ist. Mit den Signalen hat es folgendes auf sich:

Pin 2
nicht benutzt (NC).

Pin 4
nicht benutzt (NC).

Pin 6
nicht benutzt (NC).

Pin 8
Indeximpuls (Ausgang): Wenn der Motor, der die Diskette antreibt, mit seiner Solldrehzahl (geregelt) dreht, überwacht eine Fotozelle (oder ähnliches) das Indexloch der Diskette. Immer dann, wenn das Loch die Zelle passiert hat, wird über die Elektronik ein Impuls auf dieser Leitung erzeugt.

Pin 10
Drive Select 1 (Eingang): Selektionssignal für die Laufwerksnummer 1 (2, 3 und gegebenenfalls 4 bei den anderen Leitungen). Es steht somit für jedes Laufwerk eine eigene Leitung bereit.

Pin 12
Drive Select 2 (Eingang): siehe Pin 10.

Pin 14
Drive Select 3 (Eingang): siehe Pin 10.

Pin 16
Motor On (Eingang): Wenn dieser Eingang angesteuert wird, beginnt sich der Antriebsmotor zu drehen. Der Motor wird von einer Elektronik auf seiner Solldrehzahl, in der Regel 300 Upm, gehalten (geregelte Drehzahl).

Pin 18
Direction (Eingang): Dieser Eingang bestimmt, in welche Richtung der oder die Schreib-/Leseköpfe bewegt werden. Dies ist entweder in Richtung Diskettenmitte oder -rand.

Pin 20
Step (Eingang): Jeder Impuls an diesem Eingang bewegt den Kopf um eine Spur weiter und zwar in die Richtung, die mit dem Direction-Signal (Pin 18) bestimmt wird.

Pin 22
Write Data (Eingang): Hier werden die Daten,

die auf die Diskette geschrieben werden sollen, eingespeist.
Pin 24
Write Gate (Eingang):
Mit diesem Signal wird bestimmt, ob Daten von der Diskette gelesen oder Daten auf die Diskette geschrieben werden sollen.
Pin 26
Track 00 (Ausgang):
Dieses Signal wird aktiv, wenn sich der Schreib-/Lesekopf über der Spur 00 der ersten Diskettenseite, also am äußeren Diskettenrand, befindet. Das Signal ist erforderlich, um dem Controller die Möglichkeit zu geben, die Spuren mitzuzählen. Gewissermaßen eine Referenz für den Kontroller.
Pin 28
Write Protect (Ausgang):

SPUREN ZÄHLEN

um dem Controller die Möglichkeit zu geben, die Spuren mitzuzählen. Gewissermaßen eine Referenz für den Controller.
Pin 28
Write Protect (Ausgang):
Dieses Signal wird aktiv, wenn die Diskette mit einer Schreibschutzmarke versehen wurde (8 Zoll und 5.25 Zoll) beziehungsweise wenn der Sicherungsknopf auf Schutz gestellt wurde (3 Zoll und 3.5 Zoll).
Pin 30
Read Data (Ausgang):
Hier liegen die von der Diskette gelesenen Daten an.
Pin 32
Side Select (Eingang):
Bei Zweikopflaufwerken (DS) wird hier der Schreib-/Lesekopf der gewünschten Seite zugeschaltet. Bei Einkopflaufwerken wird dieses Signal nicht verwendet. Tritt es dennoch auf, wird in der Regel der Laufwerksindeximpuls abgeschaltet, was zu einer Fehlermeldung führt.
Pin 34
nicht benutzt (NC).
Pin 1 bis Pin 33
Masse der Signalleitungen.

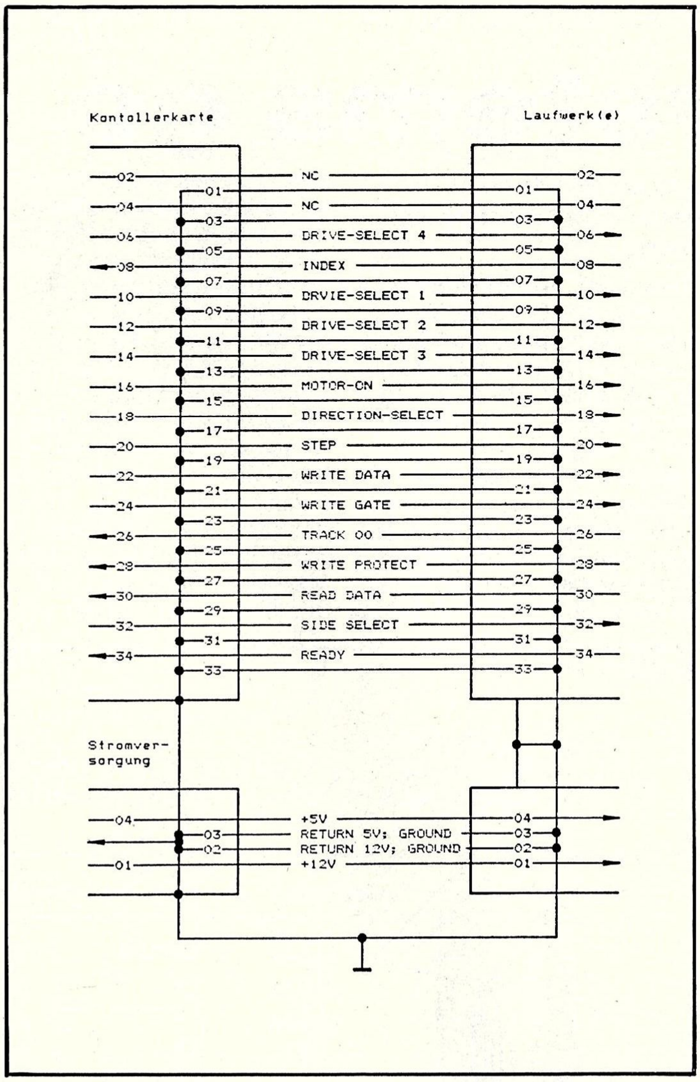


Abbildung 4: Bus-Signale

Dies ist die ursprüngliche Originalbeschaltung des Shugart-Busses der ersten Generation. Wie man sieht, sind vier der mechanisch vorhandenen Signalleitungen nicht beschaltet (Pin 2, 4, 6 und 34). Recht schnell ergänzt man die Beschaltung von Pin 6 mit dem Drive-Select-4-Signal, um die Ansteuerung von bis zu vier Laufwerken unmittelbar über den vorhandenen Bus zu realisieren. Dies erklärt, warum der Original-TI-Controller nur drei, seine Nachfolger aber vier Laufwerke bedienen können. Bei ihnen ist diese erste Buserweiterung bereits berücksichtigt und benutzt.
Später wurde der Bus um ein komplexes Signal erweitert. Es wird nur dann eine logische Eins vom Diskettenlaufwerk zum Diskettencontroller geliefert, wenn sich eine Diskette im Laufwerk befindet, sich diese mit der Nenndrehzahl dreht und auf der Signalleitung Index Indeximpulse anliegen. Dieses neue Signal, READY genannt, liegt an Pin 34 des Busses. Alle neuen Laufwerke stellen dieses Signal zur Verfügung. Da es aber mangels Vorhandensein zum Zeitpunkt der Controllerentwicklung nicht greifbar war, benutzt es der Controller auch nicht. Er hält sich nur an den Indeximpuls um festzustellen, ob sich eine Diskette im Laufwerk befindet und dreht. Das READY-Signal wird aber eigentlich immer gebraucht. Daher generiert es der Controller in seinem Inneren. Die Erzeugung dieses Signals wurde aber später auf die Platine der Laufwerke ausgelagert, um den Controlleraufbau weiter vereinfachen zu können. Am TI-DOS braucht man READY nie, obwohl es selbstverständlich verwendet werden könnte.
Des weiteren wurde der Bus um die sogenannte InUseLine ergänzt. Diese Signalleitung ist, bezogen auf das Laufwerk, ein Eingang und stellt die banale Funktion, daß beim Anlegen von logisch Eins die Leuchtdiode auf der Laufwerksfront aufleuchtet, zur Verfügung. Sie erlischt wieder bei logisch

DIE CONTROLL-LEUCHTE

Null. Nicht alle Laufwerke haben diese unmittelbare Ansteuermöglichkeit der Controll-Leuchte. Wenn Sie aber vorhanden ist, kann sie über Pin 4 des Busses bei Bedarf beschaltet werden. Üblicherweise wird die Leuchtdiode aber vom Drive-Select-Signal mitbedient.
Die von Anfang an unbenutzte Leitung an Pin 2, hat bis in unsere Tage hinein beim Original-Shugart-Bus keine Funktion zugewiesen bekommen.
Wie schon erwähnt, haben 8-Zoll-Laufwerke einen etwas breiteren Bus mit 2x25 Polen. Hier stehen noch einige Steuerein- und -ausgänge mehr zur Verfügung. So zum Beispiel das Signal "Head Load". Es versetzt in die Lage, den Kopflademagnet gezielt anzusteuern. Wird das Signal bedient, setzt der Schreib-/Lesekopf des Laufwerks auf der Diskette auf. Grundsätzlich steht dieses Signal auch bei den anderen Formaten, also 3 Zoll, 3.5 Zoll und 5.25 Zoll zur Verfügung. Hier wird es aber nicht separat an-

gesteuert, sondern ist mit dem Drive-Select-Eingang verknüpft. Wird das Laufwerk angesprochen, sprich selektiert, wird gleichzeitig der Kopf geladen. Man spart so einfach Leitungen ein. Wie schon erwähnt, sind nicht alle Laufwerke zwingend nach der Shugart-Quasinorm belegt, können aber durch Ändern von Brükken kompatibel gemacht werden. Manche Hersteller schaffen so die Möglichkeit, bei von der Norm abweichender Belegung, normalerweise am 34poligen Bus nicht direkt vorhandene Signale nach außen zu führen. Für hochspezifische Anwendungen sicher eine angenehme Sache, die bei der Verwendung von Laufwerken im Zusammenhang mit dem TI beziehungsweise dem TI-Controller keine Bedeutung erlangt. Dieser Trend, hochexotische Belegungen des Busses grundsätzlich zu ermöglichen, ist derzeit ohnehin rückläufig.

EXOTISCHE BELEGUNG

Sinn und Zweck der oben aufgeführten Signale lassen sich leicht durchschauen, wenn man einmal einen Schreib-/Lesezyklus beschreibt. Dabei wird davon ausgegangen, daß sich eine Diskette ohne Schreibschutzmarke in Laufwerk 1 befindet:
Auf den BASIC-Befehl <OLD DSK1. PROGRAMM> geschieht folgendes: Der Controller aktiviert die Drive-Select-1-Leitung und die Motor-On-Leitung. Daraufhin wartet er den Indeximpuls ab. Dann stellt er die Direction-Leitung auf Diskettenrand und legt so viel Impulse an die Step-Leitung an, bis ihm die Track-00-Leitung anzeigt, daß der Kopf sich über Spur 00 befindet. Dann stellt er die Write-Gate-Leitung auf Lesen und liest über die Read-Data-Leitung die Daten des Diskettenkatalogs aus. Gegebenenfalls fährt er mit der Direktion und Step-Leitung hin und wieder auf andere Spuren (vergleiche hierzu auch den Softwareaufbau in TI-REVUE). Sobald der Controller weiß, wo sich das mit PROGRAMM bezeichnete Programm befindet, steuert er den Kopf mit Direktion und Step über die entsprechende Spur. Durch „Synchronisation" mit den Impulsen der Index-Leitung weiß der Controller auch, wann der Kopf über welchem Sektor ist (siehe oben).
Beim Schreiben von Daten ist der Ablauf sinngemäß gleich, jedoch wird zuerst über die Write-Protect-Leitung geprüft, ob das Beschreiben der Diskette zulässig ist.
An dieser Stelle ohne Bedeutung, aber dennoch interessant zu wissen, ist, daß es heute schon vielfach intelligente Laufwerke gibt, die die Schritte über die Step-Leitung mitzählen und, sollte die Steuerung des Controllers unzulässige Werte annehmen (zum Beispiel Versuch bei 40-Spur-Laufwerken, die 41ste Spur, beziehungsweise Spur 40, anzufahren), diese Steuersignale ignorieren.
Aber dies nur am Rande. Ganz undramatisch, nicht wahr? Nun zu den elektrischen Parametern des Shugart-Busses:
Zunächst einmal arbeitet der Bus mit der sogenannten „Negativen Logik". Dabei entspricht ein elektrisches H-(oder 1)Signal einem logischen L (oder 0), und umgekehrt, ein elektrisches L (oder 0) einem logischen H (oder 1). Die Spannungen (und Hysteresen) bewegen sich in den üblichen Werten für TTL-Logic, +5V (±5%). Messen Sie mit einem Voltmeter +5V, handelt es sich also um eine logische 0, vice versa.
Die Steuerung des Busses wird von Laufwerk und Controller über Bausteine mit offenem Collecter im Ausgangstransistor wahrgenommen. In der Regel ist dies ein 74...38-TTL-Gatterbaustein. Dessen Collectoren arbeiten an einem gemeinsamen Pull-Up-Lastwiderstand. Auf der Laufwerksseite befinden sich diese Widerstände im sogenannten Terminator, der nur in einem Laufwerk eingesetzt wird. Die *Abbildung 5* zeigt die Anordnung für die *Step-Leitung*, bei Verwendung von vier Laufwerken. Im Grundzustand (alle Signale logisch 0 beziehungsweise elektrisch 1) sind die Transistoren gesperrt und an den eigentlichen Eingang gelangt über den Terminatorwiderstand die Logikbetriebsspannung (+5V). Soll nun eine logische Eins übertragen werden, wird der entsprechende Transistor kurz nach Masse durchgeschaltet. Da sein Widerstand zwischen Collecter und Emitter aber ganz erheblich kleiner ist, als der Terminatorwiderstand, gelangt nach der Kausalität von Strom und Spannung nach G. S. Ohm (Ohm'sches Gesetz), ein elektrisches Low (0) an den in Frage kommenden Eingang (=logisch H).
Der Impuls wurde weitergegeben. So wird auch deutlich, warum der Bus kurzschlußsicher ist.
Beim Verdrehen eines Steckers liegen die Signale alle zwingend an Masse. Die Logic registriert an allen Eingängen High Signal. Die Ströme die

KEIN KURZSCHLUSS MÖGLICH

fließen können, sind nicht höher als im Normalfall, da sie von den Terminatorwiderständen begrenzt werden. Der Widerstandswert pro Widerstand liegt bei maximal 150 Ohm. Daraus resultiert durch die Logicbetriebsspannung von 5 Volt ein Strom von maximal 5 Volt/150 Ohm=0.040 Ampere, sprich 40 Milliampere. Der Kreis schließt sich, wenn Sie wissen, daß in den Datenblättern der maximale Ausgangsstrom von TTL-Schaltkreisen, von Exoten abgesehen, bei eben 40 mA liegt. Diese Pull-Up-Terminatorwiderstände haben aber

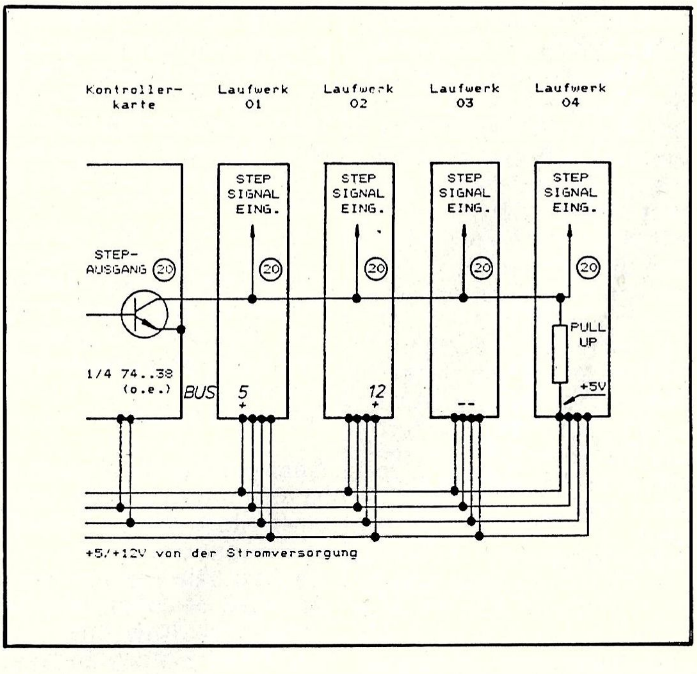


Abbildung 5: Open-Collector (OC) Schaltungsprinzip

noch eine andere Aufgabe, als die Eingänge strombegrenzend an die Versorgungsspannung zu legen. Sie schließen gleichzeitig die Leitungen des Flachband-Buskabels mit dessen dynamischen Widerstandswerten ab. Die erschöpfende Erklärung dieser Notwendigkeit würde hier allerdings zu weit führen und gehört in ein Fachbuch der Nachrichtentechnik. Soviel sei aber gesagt: Dieses Abschließen mit der Kabelimpedanz verhindert drastisch die sogenannte Leitungsreflektionen, was zu einer nicht definierten Impulszahl am Leitungsausgang führen würde.
Prinzipiell wird hier das gleiche gemacht, wie bei einem Funkgerät, welches man mit einem möglichst guten Stehwellenverhältnis an die Antenne anschließt. Elektrisch gesehen soll auch hier die Kabelimpedanz übereinstimmen. Aus diesem Grunde ist es auch nur sinnvoll, die Terminatorwiderstände in das letzte Laufwerk, an dem das Buskabel endet, einzusetzen. Da die Widerstände, wie bereits gesagt, auch die elektrische H-Spannung „liefern", muß das Laufwerk, in dem sich diese befinden, im Betrieb immer eingeschaltet sein, damit eindeutige logische Zustände am Bus herrschen. Dies wird dadurch sehr einfach, weil sich diese Widerstände in einem gemeinsamen kleinen Gehäuse befinden. Üblicherweise ist dieses Gehäuse mit dem von integrierten Schaltkreisen mechanisch identisch (DIL=DualInLine). Aber es gibt auch vereinzelt einreihige Terminatorarrays (SIL=SingleInLine). Diese Arrays sitzen auf der Laufwerksplatine in einer IC-Fassung. Daher braucht man diese Arrays, bis auf das letzte Laufwerk am Bus, nur aus ihrer Fassung zu ziehen (die Widerstandswerte sind dafür bereits ausgelegt). Das war's dann.
Auch im Controller werden die Eingangsleitungen abgeschlossen. Die entsprechenden Widerstände sind dort allerdings fest verdrahtet. Was anderes ergäbe kaum einen Sinn, weil Sie den Controller ja mit mindestens einem Laufwerk arbeiten lassen werden. Somit ergibt also die Summe aller verwendeten Signalleitungen die Anzahl der Terminatorwiderstände im gesamten Bussystem.
Im dritten und letzten Teil ergänzen wir noch Wissenswertes zum Bus und gehen außerdem auf die Stromversorgung, sowie diverses anderes ein. Man darf also gespannt sein.
Wird fortgesetzt

J. Brilmayer □

TI-Hotline
Jeden Montag
15 - 19⁰⁰
Tel.
089 / 129 8013

Alles unter Controlle?

Es gibt immer wieder Leserfragen gleichen Inhalts. Hier haben wir einige von ihnen zum Thema Disc-Controller des TI zusammengefaßt und geben Antwort.

In der letzten Zeit wurden immer wieder Fragen zum TI-Diskettencontroller gestellt. Meist geht es hierbei um das Aufrüsten auf doppelte Dichte oder auf 80 Spuren. Wir haben hier einen Brief von Herrn Daniel F. Whybrew zum Anlaß genommen, um diese und auch andere uns wichtig erscheinende Fragen ausführlich zu beantworten und hoffen, so einige Unklarheiten aus der Welt schaffen zu können.

1. Frage: *Gibt es eine Möglichkeit, den TI-Diskcontroller so umzurüsten, daß er mit doppelter Dichte arbeiten kann? Wenn nicht, warum ist dies nicht möglich?*

Antwort: Grundsätzlich ist es immer möglich, aus einer Karte für die P-Box etwas ganz anderes zu machen. Die entscheidende Frage ist dabei, in welchem Verhältnis Aufwand und Nutzen zueinander stehen, insbesondere dann, wenn es bereits fertige, käuflich zu erwerbende Karten mit den geforderten Eigenschaften gibt. Im Falle des TI-Diskcontrollers (ich nehme an, Sie meinen die Karte für die P-Box) ist die Antwort auf den ersten Teil der Frage ein klares „Jein". Damit kommen wir zum zweiten Teil der Frage – dem „Warum". Natürlich muß der Floppy-Disk-Controller WD1771-01 gegen einen WD1791-02 ausgetauscht werden. Dieser ist zwar weitgehend pinkompatibel (er hat sogar den invertierten Datenbus), doch hat diese Lösung einige Nachteile: Der bereits vorhandene Datenseparator der TI-Karte muß entfernt werden, da er ein festes Timing nur für einfache Dichte besitzt. Zwar ist dieser Datenseparator beim 1791 bereits on Chip (als PLL realisiert), doch dafür braucht der 1791 ein separates Taktsignal phasengleich zu den Lesedaten. Diesen Separator müßten Sie dazu aufbauen.
Wenn Sie auf die Pinkompatibilität keinen Wert legen, sollten Sie zum WD1773 greifen. Vorteilhaft ist hier der digitale Datenseparator, der wesentlich zur höheren Datensicherheit beiträgt. Leider ist der WD 1773 nicht einfach zu bekommen, da er zugunsten des WD1772 (im Atari ST eingesetzt) aus den Lagern vieler Händler verschwand. Der WD 1772 weist jedoch Step-Zeiten auf, die in vielen Fällen zu kurz sind (2, 3, 5 und 6 Millisekunden). Angenommen, es hätte geklappt, und Sie hätten den WD1791 mit einigen Änderungen der Platine oder den WD1773 mit sehr umfangreichen Änderungen eingebaut. Dann muß es ja auch irgendwie möglich sein, zwischen einfacher und doppelter Dichte umzuschalten. Der Floppy-Controller-Chip hat so eine Leitung, doch der TI

braucht dazu ein weiteres CRU-Latch.
Damit kommen wir zum nächsten Problem: der Betriebssystemsoftware oder DSR für den Floppy-Controller. Da der WD1791-02 nicht softwarekompatibel ist (oder zumindest nicht hundertprozentig), müssen fast alle sogenannten Level-Null-Routinen (die Routinen zur direkten Ansteuerung des Floppy-Chips) neu geschrieben werden. Dabei muß dem Umstand Rechnung getragen werden, daß nun ein Schreib- oder Lesefehler nicht immer endgültig ist, sondern vielleicht durch eine falsche Dichteeinstellung verursacht wurde. Bei einem WD1773 können Sie sich bei CorComp oder Atronic „bedienen", die 1791-Software dürfen Sie selbst schreiben.
Da hierzu die knappen acht Kilobyte des normalen TI-Controllers nicht ausreichen (Sie benötigen zum Beispiel zwei Formatier-Routinen oder wenigstens eine universelle), müssen Sie zudem ein doppelt so großes DSR-ROM (oder Eprom) vorsehen, welches auch irgendwie in den acht Kilobyte

NEUES EPROM NÖTIG

DSR-Bereich umgeblendet werden muß. Dazu benötigen Sie noch ein kleines System-RAM auf dem Controller (2114 oder besser 6116), da Sie Teile der Level-Null-Routinen bei doppelter Dichte nur im PAD-RAM ausführen können, dessen Inhalt natürlich irgendwo gesichert werden muß, damit Sie sich nicht mit einem einfachen Diskzugriff viele Ihrer schönen Programme abschießen.
Natürlich kann im Leserbriefteil der TI-REVUE keine Umbauanleitung gegeben werden. Auch und vor allem deshalb, weil man gebrauchte Controller für doppelte Dichte ab etwa 200 Mark (neue zum Beispiel CorComp für cirka 300 Mark) erwerben kann.
Sollten Sie jedoch mehr Informationen haben wollen, so können Sie über die Redaktion eine (meine) Adresse erhalten.
Da dieses Thema aber von allgemeinem Interesse ist, wurde hier in der Rubrik „Leserbriefe" etwas detaillierter geantwortet, vor allem, um den Aufwand darzustellen, den man für den, durchaus möglichen, Umbau treiben muß.
Es geht das Gerücht um, daß Ryte Data (Canada) einen Umbausatz auf doppelte Dichte für den internen TI-Controller vertreibt. Gesehen wurde ein solcher Controller noch nicht, auch der Preis liegt im Dunkeln. Hier hoffen wir auf einen Leser, der vielleicht Erfahrungen damit hat, und uns möglicherweise sogar eine Firmenadresse angeben kann.

2. Frage: *Ist DOS 80 eine solche Umrüstung? Was macht es genau? Wo kann man es bekommen, seit es das Computer Hüsli nicht mehr gibt?*

Antwort: Zuerst zum letzten Teil Ihrer Frage: DOS 80 können Sie bei jedem gut ausgerüsteten TI-Club oder jeder User-Group bekommen, wofür Ihnen in der Regel nur die Kosten für die Eproms berechnet werden sollten. Wenn man Ihnen dafür nach Ihrem Ermessen zuviel Geld abverlangt, so wenden Sie sich an die Redaktion.
Nun zum ersten Teil Ihrer Frage. DOS 80 ist eine reine Softwareangelegenheit und daher nicht in der Lage, in irgendeiner Form doppelte Dichte zu simulieren. Das liegt an der vollkommen anderen Form, in der Double-Density-Signale codiert sind. DOS 80 erlaubt dem TI-Diskcontroller für die P-Box den Einsatz von Diskettenlaufwerken mit 80 Tracks, gegebenenfalls auch doppelseitige. Bei umschaltbaren Laufwerken sind auch normale Disketten (in einfacher Dichte mit 40 Spuren) lesbar. Verwendet man aber 2 x 80 Track-Laufwerke, so verfügt man über 1440 Sektoren pro Disk, der gleichen Menge wie bei 2 x 40 Spuren in doppelter Dichte (360 KByte).
DOS 80 gibt es in zwei Versionen. Die eine Version, auf die Sie sich offenbar beziehen, arbeitet mit allen 40- oder 80-Spur-

ZWEI VERSIONEN

Formaten (1 x 40, 2 x 40, 1 x 80 oder 2 x 80), weist jedoch einige Softwareinkompatibilitäten auf, die aus der abweichenden Verwaltung einiger VDP-RAM-Byte hervorgehen. Ein zweites DOS 80 ist bei uns seit langem (etwa zwei Jahre) in Betrieb, das durch einige kleine Änderungen des Original-ROMs erzeugt wurde. Dieses DOS 80 ist vollkommen software-neutral, kann auf die kurzen Step-Zeiten der 80-Track-Laufwerke angepaßt werden, und hat als einzigen Nachteil keine Möglichkeit, 2 x 40 Spuren zu verwalten.
Mit einem kleinen DIAGS-Kommandofile ist aber auch damit ein Umkopieren möglich. Wir werden in einer der folgenden Ausgaben der HCA eine Liste der Patches im Original-ROM abdrucken. Die DOS 80 ROMs ersetzen U16 und U17 des Controllers.

3. Frage: *Kann man einen TI99 und den Geneve parallel zueinander laufen lassen, vorausgesetzt, daß man getrennte Monitore benutzt?*

Antwort: Nach dem derzeitigen Kenntnisstand der Redaktion und dem offiziellen Entwicklungsstand des Myarc 9640 Geneve ist es nicht möglich, beide Systeme parallel (Sie meinen wahrscheinlich die gemeinsame Benutzung der P-Box) als quasi Multiprozessorsystem zu benutzen. Der Geneve arbeitet nur dann, wenn neben anderen Karten auch das Flex-Cable-Interface aus der Box entfernt wurde. Doch nur mit diesem Interface kann der TI99 irgend etwas in der P-Box anstellen. Doch wie oben anhand des Diskcontrollers bemerkt, kann man sich schon etwas einfallen lassen.
Der Geneve ist so lange ein in sich abgeschlossenes System, wie er nicht auf externe Hardware-ware wie etwa Myarc-RAM-Disks oder Diskcontroller zugreift. Er muß also Vorrang haben, wenn er an den Bus der P-Box

GESCHLOSSENES SYSTEM

will. Eine Synchronisation mit dem 99/4A ist jedoch nicht möglich, da der Geneve keine Möglichkeit bietet, die CPU in der Art der Ready-Leitung anzuhalten. Also muß der TI00/4A gestoppt werden.
Zwar kann man die Adreßbustreiber des Flex-Cables abschalten, auch der Datenbus ist abschaltbar. Doch leider muß dies auch beim Geneve möglich sein, und dazu muß man Näheres über das Gate-Array wissen. Doch gibt sich Myarc hier besonders zugeknöpft. Und schließlich muß in jedem Fall irgendwo an der Vierfach-Multiplayer-Platine des Geneve gelötet werden. Da aber in der zweiten Lage auch Leitungen liegen und es zudem in Deutschland keine autorisierte Myarc-Vertretung gibt, wo man die abgeschossene Maschine reparieren lassen kann, muß von einem solchen Eigenbau dringend abgeraten werden!
Kaufen Sie sich besser eine gebrauchte P-Box (mit Controller und Flex-Cable-Interface im Schnitt für

400 Mark zu haben), ein CPS99 von Atronic, ein MPS 500 von Myarc oder das CorCom-Micro-Expansion-System und versuchen Sie eine Kopplung über RS232.

4. Frage: *Wo bekommt man ein Info über die Festplatte für den TI? Was leistet sie?*

Antwort: Zuletzt wurden Infos über den Myarc-Harddiskcontroller in Köln 1986 verteilt. Die Firma Myarc liefert derzeit eine neue Version der Karte aus (man hat es noch nie erlebt, daß ein Myarc-Produkt gleich zu Anfang in Ordnung war und nicht etliche Revisionen kamen), an die bis zu vier Harddisks zu je maximal 20 Megabyte Kapazität und bis zu vier Floppy-Disks angeschlossen werden können. Angesprochen wird die Harddisk über WDS1. bis WDS4., wobei durch weitere Punkte Subdirectories ähnlich MS-DOS aktiviert werden können.
Die Winchester ist auf DSR-Level 2 voll kompatibel zu bestehender Software, nicht jedoch darunter. Das ist aufgrund des abweichenden DOS auch nicht anders möglich.
Leider liegen uns die Informationen zum neuen Controller nicht vor (Myarc liefert nur den Controller als Hersteller, Harddisks sind jedoch von Fremdherstellern erhältlich), doch sicher findet sich ein Leser, der hier weiterhelfen kann.

5. Frage: *Was genau ist „Genial Traveller", was kostet er und wo kann man in den USA hinschreiben, um ihn zu bekommen? 20 Mark pro Diskette scheint mir zu teuer zu sein.*

Antwort: Aus dem Text Ihrer Frage ist zu entnehmen, daß Ihnen der TRAVelER (kein Schreibfehler!) für 20 Mark angeboten wurde. Bitte teilen Sie uns mit, wer dieses Angebot gemacht hat, da es auch Aufgabe einer User-Zeitschrift ist, auf solche Praktiken warnend hinzuweisen. Der TRAVelER geht auf eine Idee von Barry Traver zurück (daher die etwas eigenartige Großschreibung), nach der sich einige amerikanische Programmierer zusammengetan haben, um eine Art Diskettenzeitschrift aufzuziehen. So vertreiben Sie Programme wie Archiver, Systex etc. auf Disketten mit dem Vermerk "Let me travel", womit Name und Funktion endgültig klar sind.
Eine Kontaktadresse wäre zum Beispiel die Chicago TI-User Group, zu deren Autoren Barry Traver zählt. Was nun den Preis betrifft, so ist die Sachlage etwas komplizierter. TRAVelER ist de facto *nicht* Public Domain, da sich die Autoren das Copyright erhalten wollen (Public Domain ist in gewisser Weise gleichbedeutend mit völligem Verzicht auf alle Ansprüche). 20 Mark sind, bei Einsendung einer formatierten Leerdiskette und adressiertem und frankiertem Rückumschlag jedoch Raubkopiepraktiken. In den USA sind bei solchen Bestellungen an User-Groups im allgemeinen 10 US-Dollar fällig, bei uns also DM 10,– (wegen des Geldwertes). Firmen haben nach dem Willen der Freeware-Autoren dazu kein Recht, sie dürfen kein Geld verlangen, das Programm aber bei anderen Bestellungen kostenlos beilegen.
Da dieses Thema, Freeware, Shareware, Public Domain und was es noch an Kunstbezeichnungen dafür gibt, von großem allgemeinen Interesse ist, wird sich ein Beitrag in einer folgenden TI-REVUE damit ausführlich befassen.

6. Frage: *Gibt es eine Möglichkeit, mit dem TI-Writer mehr als 80 Zeichen pro Zeile zu schreiben?*

Antwort: Ohne umfangreiche Änderung des TI-Writer ist es nicht möglich, über die 80 Zeichen zu kommen. Doch es gibt ja die Alternative WriterEASE von CorComp. Dieses hochinteressante Programm erlaubt Zeilenlängen von bis zu 254 Zeichen, womit Sie Ihren Drucker voll auslasten können. Bezugsquelle ist TEXCOMP, P.O. Box 33084, Granada Hills, CA 91344, der Preis beträgt 49,95 US-Dollar. Das Programm verfügt daneben über einen (leider nicht lernfähigen) Spell-Checker und sehr umfangreiche Hilfstexte.

Christopher Winter □

Club-Ecke

ERRORFREE

Nachdem sich in der letzten Ausgabe mit dem TI-Club Baunatal eine User-Gruppe aus dem nordhessischen Raum um Kassel vorgestellt hat, folgt heute ein Club von TI-Anhängern aus der Gegend um Stuttgart. Auch hier bemühen sich User des TI99/4A darum, Kontakte zu anderen Freunden dieses Rechnertyps zu bekommen. Das Ziel, eine Liste aller heute noch existierenden TI-Zusammenschlüsse zu erstellen, ist unserer Meinung nach sehr zu begrüßen. Auch wir haben uns ja das Ziel gesetzt, unseren Lesern in jeder Ausgabe einen (oder auch mehrere) dieser Clubs vorzustellen. Vielleicht finden Sie so auch in Ihrer Nähe einige TI-Freaks, die Ihnen bei Fragen oder bei Problemen weiterhelfen können. Doch nun zu der versprochenen Vorstellung des TI-CLUB ERRORFREE.

Am 29.8.1987 fand die erste Mitgliederversammlung des TI-Club ERRORFREE statt. Hierbei wurde eine Satzung verabschiedet. Ebenfalls wurde einem jährlichen Clubbeitrag von 18 Mark zugestimmt. Unser Clubjahr beginnt am 1. Oktober und endet am 30. September des folgenden Jahres. Bei einem Beitritt innerhalb eines Clubjahres wird der Clubbeitrag anteilig der verbleibenden Monate berechnet. Der Clubbeitrag wird ausschließlich zur Kostendeckung erhoben.
Der Sinn und Zweck des TI-Club ERRORFREE ist es, den TI99/4A zu erhalten sowie bei Soft- und Hardware-Problemen seinen Mitgliedern zu helfen.
Obwohl durch Systemwechsel einige Mitglieder ausgeschieden sind, hat der TI-Club ERRORFREE wieder 26 Mitglieder.
Auf diesem Weg möchte ich alle TI-Clubs, TI-Workshops und User-Groups auffordern, an die Kontaktadresse den Clubnamen und Kontaktadressen zu senden, damit eine Liste von den noch aktiven Clubs erstellt werden kann. Um unsere Portokasse zu entlasten, bitte Rückporto beilegen.

Weitere Informationen können bei folgender Kontaktadresse angefordert werden:

TI-Club ERRORFREE
c/o Manfred Kraus
Saarstraße 27
7254 Hemmingen □

MG DISKASSEMBLER

Assembler rückwärts

Mit der Besprechung dieses jüngsten Produktes der Millers Graphics Softwareschmiede wird die Beitragsreihe über die bemerkenswerten Programme dieser Firma beendet. Doch keine Angst – sobald das neue Extended BASIC-Modul-Version120 von MG eingetroffen ist, berichten wir in HCA darüber.

DISKASSEMBLER ist, wie auch die anderen beiden, DIAGS und EXPLORER, ein Programm, das nichts bringt, was im Grundsatz neu wäre.

Doch die Realisierung zeigt auch dieses Mal, daß man bei MG eine enorm anwenderfreundliche Synthese aller Merkmale bereits vorhandener Programme (hier Diskassembler; davon gibt es für den TI eine ganze Menge) fand und zudem einige neue Optionen einfügte.

Das soll ein kleiner Überblick verdeutlichen:

- DISKASSEMBLER kann entweder Speicherinhalte disassemblieren oder direkt von Diskette disassemblieren, also direkt im File.
- Speicherinhalte kann so gut wie jeder Disassembler analysieren, und wer von Diskette arbeiten will, der kann den „Universal Disassembler" von Rene LeBlanc verwenden, der als Freeware verfügbar ist.
- Aber DISKASSEMBLER benötigt als einzige Information den Filenamen, wogegen man sich beim Universal Disassembler erst mühsam die Sektoren des Files zusammensuchen muß.
- DISKASSEMBLER kann das Ergebnis seiner Arbeit als File zur Verfügung stellen.
- Das kann der Weiland Disassembler V2.1 auch, der ebenfalls als eine Art Freeware verfügbar ist. Doch der kann keine Files disassemblieren.
- DISKASSEMBLER erlaubt das blockweise Disassemblieren eines beliebigen Maschinencodes. Das kann sonst kein anderes Programm. Bei allen anderen Programmen muß das in Handarbeit geschehen, indem abwechselnd Start- und Endadressen neu eingegeben und dazwischen Daten als Hexdump aufgelistet werden. DISKASSEMBLER macht das nach Vorgabe der Bereiche selbständig.

Soweit dieser kurze Vergleich, schließlich soll das eine Programmbeschreibung und keine Disassemblerübersicht werden.

Was ist ein Disassembler?

Doch halt! Die Kernfrage wurde bisher noch nicht beantwortet. Was ist denn überhaupt ein Disassembler?

Die Antwort klang in der Überschrift schon an und auch die Bezeichnung dieser Art Programme mag als Hinweis dienen.

Ein Disassembler macht die Arbeit eines Assembler rückgängig, indem er aus einem Maschinenprogramm wieder einen Quellcode erzeugt, der, je nach Qualität des Disassemblers, mehr oder weniger ähnlich dem Quellcode ist, den der Autor beim Assemblieren vorliegen hatte.

Normalerweise sieht man ja nur, was ein Programm macht, also den Output. Ein Disassembler zeigt, *wie* es gemacht wird. Natürlich muß man zur Interpretation dessen, was ein Disassembler liefert, selbst in der Lage sein, Assemblerprogramme zu schreiben. Doch dadurch, daß einem ein Disassembler den Einblick in das Know-how anderer Programmierer erlaubt, kann man den einen oder anderen Kniff lernen. Und genau das ist m.E. die hauptsächliche Existenzberechtigung eines solchen Programms.

Leistungsmerkmale

Das hat wohl auch Thomas S. Freeman gedacht, als er DISKASSEMBLER schrieb, denn das Programm weist viele Punkte auf, die das Arbeiten mit ihm erleichtern.

So erkennt DISKASSEMBLER vollkommen selbständig, welche Art von File vorliegt (Tagged Object Code oder Memory Image), wie lang es ist (bei Standard-Typen) und inwieweit der Maschinencode an feste Adressen gebunden ist.

Er erlaubt die Spezifikation von Datenfeldern, die keinen Maschinencode enthalten und daher nicht disassembliert werden müssen sowie die Angabe von besonderen Unterprogrammaufrufen, denen Datenworte folgen. Bei diesen erscheinen dann die DATA als Daten und nicht als wenig sinnvolle Befehle.

Daneben erlaubt das Programm die simultane Ausgabe des Ergebnisses auf Bildschirm, Drucker und Diskette (als File, das in vielen Fällen direkt wieder assembliert werden kann). Dabei wird die Länge eines jeden Files auf maximal 65 Sektoren begrenzt und danach ein neues geöffnet. Auch macht es nichts, wenn die Diskette voll wird – das letzte File wird korrekt geschlossen und das Einlegen einer neuen Diskette abgewartet – kein Datenverlust!

Doch das wichtigste ist: DISKASSEMBLER ist ein Zwei-Pass-Disassembler. Das bedeutet, daß er sich zweimal um den Inhalt eines Files oder Speicherbereiches kümmert. Dabei „merkt" er sich gewisse Parameter (genau genommen baut er eine Symboltafel ähnlich einem Assembler auf), mit deren Hilfe er im zweiten Pass (Durchlauf) richtige Labels vergibt. Damit ist das Ergebnis seiner „Arbeit" näher am Original des Source-Codes als das irgendein anderer Disassembler erlaubt.

Aufgrund dieser Eigenschaft wird DISKASSEMBLER als „Symbolischer Diskassembler" bezeichnet,

weil er in der Lage ist, auf Speicherinhalte nicht in direkter Weise, sondern eben über Symbole hinzuweisen, eine wichtige Voraussetzung zum Erreichen eines relokatiblen Codes.

Aber auch DISKASSEMBLER vollbringt keine Wunder. Denn daß der produzierte Quellcode fast immer direkt und ohne Fehler assembliert werden kann, hilft ja gar nichts, denn dann ist man ja so weit wie vorher. Wer irgend etwas ändern will, kommt um das Verständnis dessen, was DISKASSEMBLER liefert, nicht herum. Dazu gehört nach wie vor einige Erfahrung in der Programmierung von TMS9900-Maschinensprache.

Bei der Gelegenheit einige Worte zu dem zweideutigen Begriff „Assembler". In der deutschen Sprache gibt es ein passendes Beispiel, den „Widerstand". Einmal ist es eine Sachbezeichnung und dann eine Eigenschaft. Ähnlich beim Assembler. Gelegentlich wird die Maschinensprache einer beliebigen CPU als „Assembler" bezeichnet, üblicherweise zusammen mit der vorangestellten Bezeichnung der betreffenden CPU. Um das zeitraubende Bitgefummele der echten Maschinenbefehle zu umgehen, gibt man nun jedem Operationscode (Op-Code) der CPU einen Namen, das sogenannte Mnemonic. Mit diesen Namen und deren Zusätzen (der Syntax) schreibt man nun ein Programm in einer Sprache, die „Assembler" genannt wird. Um diese Sprache auf das Bit-Niveau zu bringen, verwendet man einen „Assembler" – einen „Zusammenfüger". Genaugenommen liegt aber diese Programmierung in Assembler schon einen Schritt über dem Prozessorniveau, so daß der Assembler, der Assemblerbefehle assembliert, eigentlich „Compiler" heißen müßte. Da so aber nur solche Programme heißen, die eine noch weiter vom Maschinenniveau entfernte Hochsprache wie zum Beispiel BASIC, PASCAL oder C in Maschinensprache übersetzen, blieb es beim doppeldeutigen Assembler. Das soweit zur Begriffsbestimmung.

DISKASSEMBLER wird, wie üblich, als Diskette im Format 1S geliefert, also dem Format, was wohl alle TI-Systeme lesen können.

Der Loader von DISKASSEMBLER akzeptiert alle Controller, von Atronic über CorComp und Myarc bis TI-Original. Für den Geneve ist eine Sonderversion des Programms nötig, unabhängig vom eingesetzten Controller.

Für das CPS99 müssen ähnliche Patches im Loader vorgenommen werden, wie das beim EXPLORER notwendig war, damit das Programm startet.

Auf der Diskette befinden sich lediglich drei Files. LOAD ist der Loader für den automatischen Start aus Extended BASIC, DISKCONFIG ist das File, in dem sich die Optionen des Hauptprogramms befinden und DISKASSEM ist der universelle Loader für Editor/Assembler, Mini-Memory und Extended BASIC.

Aus unerfindlichen Gründen (nicht, daß sie nicht zu finden seien, sondern das Warum ist unerfindlich) springt der Loader bei einem TI mit PAD-RAM-Erweiterung direkt zum Titelbild. Dieses sollte beim Booten abgeschaltet werden.

Wie üblich liegt ein ausführliches Manual bei (58 Seiten), welches auf 34 Seiten eine Programmanleitung bekannter Güte beinhaltet. Daran anschließend folgen die Anhänge, zuerst ein Anwendungsbeispiel von DISKASSEMBLER anhand der SAVE-Utility des EA-Moduls (Seiten 34 bis 42), zwei Seiten Tips, wie man zu lange Programm-Files in zwei kleine zerlegt und 10 Seiten Tabellen mit Speicherbelegungen, wie man sie unter den drei verschiedenen Modulen vorfindet.

Nachdem der Loader gestartet wurde, lädt das Programm innerhalb weniger Sekunden. In dieser Zeit erscheint eine Copyright-Meldung mit Firmen- und Autorenname sowie dem Copyright-Datum. Dieser Warte-Bildschirm wird dann vom ersten Arbietsbildschirm abgelöst. Hier wird man zur Eingabe eines Filenamens aufgefordert. Löscht man diesen mit FCTN 3, so kann ein Speicherbereich disassembliert werden. Dabei ist DISKASSEMBLER genauso kapriziös wie EXPLORER – alles, was in seinen Bereich fällt, wird nicht bearbeitet. Wird also Ihre Eingabe von Start- und Endadresse ignoriert, so fühlte sich das Programm „bedroht", weil Sie es selbst disassemblieren wollten.

Da DISKASSEMBLER am oberen Ende der Speichererweiterung „residiert", können andere Programme vorher geladen und diese dann dierekt im Speicher untersucht werden, vorausgesetzt, sie belegen nicht den Speicherbereich von >3000 bis >3FFF und >D000 bis >FFFF. Das ist vor allem dann empfehlenswert, wenn diese Programme externe Referenzen besitzen wie DSRLNK, VSBW, KSCAN etc.

Da diese vom Loader des jeweiligen Moduls rückwärts aufgelöst werden, spart man sich so eine Menge Zeit. Näheres folgt weiter unten.

In der unteren Bildhälfte des ersten Arbeitsschirmes befinden sich zusätzliche Informationen zum Objekt der Untersuchungen.

TYPE
Gibt bei File-Dissassemblierungen den Filetyp an.

CODE
Zeigt an, ob es sich um komprimierten, unkomprimierten oder Memory-Image-Code handelt.

LENGTH
Zeigt die Anzahl Byte im Programm an, die vom Loader an beliebige Stellen des Speichers geschrieben werden können (Relocatable Code). Eine Null hier sagt aus, daß das Programm an feste Adressen gebunden ist.

LOADER
Gibt an, ob das File mit einem normalen Loader irgendeines Moduls geladen werden kann, oder ob dafür ein weiteres Anwenderprogramm nötig ist (Angabe Standard oder Custom).

LOADS AT
Gibt die Ladeadresse an. Der Zusatz RORG gibt an, daß die Ladeadresse im Prinzip beliebig ist und nur von DISKASSEMBLER willkürlich auf die erste freie Standardadresse gelegt wurde (so, wie es jeder moduleigene Loader tut). AORG gibt wieder die Bindung an einen festen Adreßbereich an.

SECTORS
Taucht nicht bei DIS/FIX-Files auf. Gibt ansonsten die Länge eines Memory-Image-Typ-Files (Program-File) an, so wie sie im File-Descriptor-Sektor des Files steht.

OPTIONS
Damit kann die Arbeitsweise von DISKASSEMBLER festgelegt werden. Verfügbare Optionen:

R Wie beim Assembler – stellt ein 'R' vor jede Registernummer. Wird diese Option nicht gewählt, so können auch Labels Namen wie zum Beispiel 'R5' oder 'R10' erhalten.

B Subtrahiert gegebenenfalls einen BASIC-Screen-Offset von Texten, damit diese lesbar werden. Betrifft nur Programme, die aus einem BASIC heraus benutzt werden.

T Gibt der Anzeige von Text-Statements den Vorzug gegenüber den DATA. Siehe bei der Besprechung des Ausgabeformats.

D Ist nur bei Untersuchung von Speicherbereichen notwendig und gibt an, daß eine DSR untersucht werden soll. Damit wird bewirkt, daß simultan zum Auslesen der spezifizierten Speicherbereiche bestimmte CRU-Bit gesetzt werden können.

X Normalerweise vergibt DISKASSEMBLER an Adressen, die nicht direkt im Bereich des spezifizierten Speichers oder des Files liegen, keine Labels. Wird 'X' aktiviert, so macht er es trotzdem und fügt eine Equate-Liste der externen Referenzen an.

V Bei externen Referenzen, wie oben angeführt (VSBW etc.), werden diese vom moduleigenen Loader rückwärts aufgelöst. Liegen derartige Referenzen vor, so muß diese Option beim zweiten Pass aktiviert werden, so daß die Referenzkette klar erkennbar wird. Ansonsten würden an den entsprechenden Stellen nur Nullen auftauchen. Ob solche Referenzen existieren, wird nach dem ersten Pass angezeigt (out of order origins).

M Zeigt an, daß ein Myarc-Diskcontroller benutzt wird. Da sich dieses Gerät nicht an die Konventionen zur Benutzung des VDP-RAMs hält, muß dann an anderer Stelle nach den File-Informationen gesucht werden, die DISKASSEMBLER benötigt (genauer im System-RAM des Controllers ab >5000).

G Übersteuert die Option 'M' und zeigt an, daß die Files von einer RAM-Disk geholt werden. Viele der vorhandenen RAM-Disks benutzen den VDP-Puffer überhaupt nicht, so daß DISKASSEMBLER den File-Status getrennt mittels der Level-1-Routine >14 bestimmen muß.

Alle diese Optionen können im File DISKCONFIG festgelegt werden, wobei M und G tunlichst in diesem File festgelegt werden sollten, da diese Optionen bereits beim Laden eines Files aktiv sind und man sich so Fehl-Ladevorgänge spart..

Es ist aber möglich, daß man erstmal keinen Filenamen eingibt und sich bis zu den Optionen vorarbeitet und diese dann korrekt einstellt. Mit FCTN 9 kann dann erneut ein Filename eingegeben werden.

DISKCONFIG kann zum Beispiel mit dem EA-Editor bearbeitet werden. Es ist immer genau drei Zeilen lang. In der ersten Zeile steht die Einstellung der Bildschirmfarben. Sie werden als Hex-Ziffern eingegeben, wobei die erste Ziffer für die Vorder- und die zweite für die Hintergrundfarbe steht, zum Beispiel F1 für Weiß auf Schwarz oder F4 für Weiß auf Dunkelblau, dem Standard-Wert des Programms. In der zweiten Zeile können die Optionen eingegeben werden. Sie erscheinen später genau so im Options-Feld des Hauptprogramms. Die dritte Zeile enthält den Namen des Druckers. Dieser darf maximal 38 Zeichen lang sein. Laden Sie doch einfach das vorhandene File, womit Sie ein Beispiel haben.

Die Bedienung

Wie bereits mehrfach erwähnt, erlaubt DISKASSEMBLER die Disassemblierung von Speicher- und Fileinhalten.

Wird im ersten Bildschirm ein Filename eingegeben, so wird dieses File disassembliert. Wird kein Filename eingegeben, so wird ein anzugebender Speicherbereich im CPU-Adreßraum disassembliert.

GRAM/GROM oder VDP können nicht untersucht werden. Da dort aber auch keine Maschinenprogramme zu finden sind, ist diese Einschränkung tolerierbar.

Will man DISKASSEMBLER verlassen, so muß noch innerhalb der Filenamenzone CTRL= gedrückt werden. Das Programm kann nur so verlassen werden.

Ansonsten stehen folgende Funktionen zur Verfügung (Reihenfolge wie im Handbuch):

FCTN 1, 2 wie üblich zum Löschen und Einfügen eines Zeichens. FCTN 3 zum Löschen einer Eingabezeile, FCTN 4 bricht komplett ab und führt zur Eingabe des Filenamens, FCTN 6 zeigt den folgenden Bildschirm an (es gibt insgesamt 4), FCTN 9 führt schrittweise jeweils einen Bildschirm zurück und CTRL 1 läuft durch die verschiedenen Möglichkeiten von Bildschirm-Vorder- und Hintergrundfarben.

Daneben sind die Cursortasten aktiv, mit denen man zwischen den Eingabefeldern herumfahren kann.

Disassemblieren von Speicherinhalten

Wird die Filenameneingabe mit einem Leerstring und ENTER verlassen, so erscheint ein fast leeres Informationsfeld, unter dem Start- und Endadresse eingegeben werden können. Dabei ist die Endadresse, wie auch bei allen anderen Hexadezimal-Eingaben, nicht inklusive. Wurde im Optionsfeld ein 'D' eingegeben (DSRs disassemblieren), so erscheint zusätzlich eine Zeile mit CRU-Adressen. Dabei kann eine Softwarebasis-Adresse angegeben werden, an der das CRU-Bit 0 (relativ) gesetzt wird, der Speicherbereich in einen Puffer geholt und dieses Bit anschließend wieder zurückgesetzt wird. Es können bis zu vier

Bitte lesen Sie weiter auf S. 51

Fortsetzung von Seite 32

Bit simultan geschaltet werden. Nullen als Adresse werden ignoriert. So muß zum Beispiel für die Untersuchung des Diskcontrollers als Adresse>1100 eingegeben werden, bei einer Startadresse>4000 und einer Endadresse von>6000, effektiv also >5FFE. Für die zweite ROM-Bank muß als zweite CRU-Adresse >1116 eingegeben werden.

Wurden alle Eingaben gemacht oder FCTN 6 gedrückt, so erscheint der zweite Bildschirm. Ab jetzt unterscheiden sich Speicher- und File-Disassemblierung im weiteren Ablauf nicht mehr.

Disassemblieren von Diskettenfiles

Nachdem der Name eines existenten Files eingegeben wurde (etwa DSK1.DISKASSEM – das geht), wird geprüft, von welchem Typ das File ist. Program-Files werden direkt geladen, sofern sie nicht länger als 47 Sektoren sind. DIS/FIX-Files werden nur satzweise geladen.

Unterhalb der Informationszone erscheinen die Standard-Adressen der Loader für die drei Module, mit denen DISKASSEMBLER geladen werden kann. Sofern das File nicht adreßgebunden ist, setzt DISKASSEMBLER einen Default-Wert für LOADS AT ein, der jedoch manuell beeinflußt werden kann. Bei Program-Files kann auch deren Länge manipuliert werden (soweit, wie DISKASSEMBLER betroffen ist), was aber nicht immer sinnvoll ist. Die Vorgabewerte können mit ENTER akzeptiert werden, FCTN 6 führt zum folgenden Bildschirm.

Bildschirm 2 – Block-Disassemblierung

Hier können Sie innerhalb des Adreßbereiches des Files oder Speicherbereichs bestimmte Zonen als Daten- beziehungsweise Textblöcke deklarieren, die nicht explizit disassembliert werden, sondern mit vorangestelltem 'TEXT' oder 'DATA' erscheinen. Das spart enorm Platz auf der Diskette, die das Ausgabefile enthalten soll und beschleunigt die Disassemblierung. Eingegeben werden Start- und Endadresse eines jeden Blocks, die Endadresse wiederum nicht inklusive.

Mit ENTER gelangt man von Feld zu Feld, FCTN 6 führt zum dritten Bildschirm.

Bildschirm 3 – Subroutinen mit Datas

Eine recht elegante Art der Parameterübergabe bei TMS9900-Maschinencode ist die, hinter einem Subroutineaufruf mehrere Datenworte als Konstanten zu übergeben, auf die dann mittels Register 11 oder 14, je nach Aufruf, zugegriffen werden kann. Diese Daten können einen Disassembler recht ordentlich durcheinander bringen, weshalb DISKASSEMBLER darauf achtet, diese DATA zu übergehen.

Dazu muß man sie ihm jedoch erst angeben. Dies wird beim ersten Durchlauf noch nicht bekannt sein, doch dient dieser erste Lauf ohnehin nur dem Überblick, zudem kann er gegebenenfalls wiederholt werden. Eingegeben werden kann die Adresse der Subroutine und die Anzahl der DATA, die dem Aufruf folgen. Achtung: Anzahl rechtsbündig schreiben! FCTN 6 führt auch hier weiter, zum vierten Bildschirm.

Bildschirm 4 – Ausgabeoptionen

Dies ist die letzte Station, bevor DISKASSEMBLER mit der Arbeit beginnt. Die hier erlaubten Tastenfunktionen werden angezeigt.

FCTN 7 (Select Printer Output) aktiviert die Druckerausgabe. Vorher wird noch der Name des Druckers abgefragt, mit dem Default-Wert aus DISKCONFIG. Wird hier der Name eines Diskfiles angegeben, so wird ein einziges File erzeugt, im Gegensatz zu FCTN 8 (Select Output Device), wo die Filelänge überwacht wird und mehrere, maximal 65 Sektoren lange Abschnitte erzeugt werden. Es empfiehlt sich, als Filename des Output Device einen Namen mit einem 'A' am Ende zu wählen. DISKASSEMBLER inkrementiert bei jedem Öffnen eines neuen Files den letzten Buchstaben, was beim Start mit 'A' immerhin 26 vernünftige Namen erzeugt. Hier darf kein Druckername eingegeben werden, da dann die Namensänderung ungültige Druckernamen erzeugen würde.

Die gute alte Quit-Taste ist hier entschärft und schaltet nur die Geschwindigkeit um, mit der disassembliert und damit gescrollt wird.

ENTER startet die Disassemblierung.

Der Output von DISKASSEMBLER

Angenommen, Sie haben diese Anleitung bis hier hin befolgt. Dann beginnt DISKASSEMBLER mit dem ersten Durchlauf (Pass) seiner Untersuchungen. In diesem wie auch im zweiten Pass besteht der Output aus sechs Spalten.

Ganz links stehen die Adressen, denen jedoch 'LN' vorangestellt wurde. Damit sind diese Adressen gültige Labels für den Assembler des EA-Moduls. Ein derart aufgebautes File kann prinzipiell direkt anschließend wieder assembliert werden. Da jedoch jede Zeile ein Label erhält, wird die Symboltafel des Assemblers sehr schnell überlaufen. Das Verfahren ist also nur bei kurzen Abschnitten sinnvoll.

In der zweiten Spalte von links stehen die Op-Codes (Mnemonics), welche die von der CPU auszuführenden Befehle darstellen. Je nach Option 'T' (an oder aus) folgt die ASCII- oder DATA-Interpretation der zweiten Spalte, also entweder Spalte 3 DATA, Spalte 4 TEXT oder Spalte 3 TEXT und Spalte 4 DATA. Spalte 5 (zweite von rechts) zeigt an, wieviel relokatible Wörter sich in der Anzeigezeile befinden (RL1 oder RL2). Diese Information ist nur bei eingehenden Änderungen im untersuchten Programm von Bedeutung und deutet auf Label-Bezüge innerhalb relokatibler Segmente hin. Ganz rechts steht in Spalte 6 die laufende Adresse, bezogen auf den Anfang des Files/Speicherbereichs.

Während nun diese Informationen an Ihnen vorbeilaufen, stehen folgende Funktionen zur Verfügung: FCTN 1, 2 schalten die Option 'T' vorübergehend aus und an, übersteuern also diese Option während der Ausgabe. FCTN 4 stoppt die Ausgabe in Richtung Drucker (Off-Line–Cancel), FCTN 5 schaltet um zwischen der Anzeige der Spalten 1 bis 3 und den Spalten 4 bis 6. FCTN 7 erlaubt das zwischenzeitliche Schließen bzw. Öffnen des Druckerkanals, so daß bestimmte Segmente nicht ausgedruckt werden. FCTN 8 macht dies analog mit dem Ausgabefile. FCTN 9 bricht die Arbeit ab, wobei alle Files geschlossen werden und zu Bildschirm 1 gesprungen wird.

FCTN 0 erlaubt es, quasi mittendrin den momentan angezeigten Bildschirm auszudrucken oder abzuspeichern (Ausgabemedium wie FCTN 7). Erscheint die Meldung 'Screen printed', so kann durch Druck auf ENTER die Ausgabe auf dem nun zugeschalteten Gerät fortgesetzt werden; FCTN 0 erneut gedruckt, schließt die Ausgabe wieder. FCTN= schaltet wie angesprochen das Arbeitstempo um und CTRL 1 erlaubt die Einstellung der Bildschirmfarben. Den ersten Pass schickt man am besten über den Drucker.

Der zweite Durchlauf

Wurde der erste Durchlauf normal beendet (kein Abbruck durch FCTN 9), so wird nach Druck auf ENTER der zweite Pass aktiviert. Zuerst sieht es so aus, als ginge alles nochmal von vorne los, denn die vier Arbeitsbildschirme werden erneut durchlaufen. Doch wird das File sofort wieder geladen. Auch alle Einstellungen auf Bildschirm 2 und 3 sind erhalten geblieben. Sie können diese jedoch ändern, sollte sich beim ersten Pass irgendeine Änderung als nötig erwiesen haben. Auch die Optionen können noch geändert werden. Normalerweise reicht es, dreimal FCTN 6 zu drücken. Nach Spezifikation von Drucker und Ausgabefile beginnt der zweite Pass. Nun tauchen in Spalte 1 und 2 Labels auf, immer mit zwei Buchstaben oder einem Buchstabe und einer Zahl. Diese Namen werden von vorne nach hinten vergeben, das heißt, sie laufen in der Reihenfolge, in der die Referenzen auf die Labels auftauchen. Damit sieht der Output schon fast wie ein richtiger Source-Code aus, bis eben auf die Namen der Labels. Der zweite Pass eignet sich für die Ausgabe in ein Diskettenfile.

Einige Tips zum Arbeiten mit DISKASSEMBLER

Ein Programm wie DISKASSEMBLER stellt verschiedene Anforderungen. Wer es benutzt, um einfach nur Programme von anderen Autoren zu sehen, der kommt mit einem guten Grundlagenwissen in Assembler aus. Wer aber das analysierte Programm in eine Form bringen will, daß eigene Ideen eingebaut werden können, der muß fortgeschrittene Assemblerkenntnisse besitzen. Um zum Beispiel DATA hinter BL und BÖWP korrekt zu zählen, muß erkannt werden, welche Befehle denn wirklich unsinnig sind, wie oft über R14 oder R11 mit Autoinkrement zugegriffen wird und wieviele Byte-Zugriffe dabei sind. Dann kann es passieren, daß Konstanten mit Werten, die rein zufällig im Adreßbereich des Programms liegen, als Labels erkannt werden. Wird das nicht berücksichtigt, ist ein solches Programm nach Änderungen in der Regel nicht mehr lauffähig, auch wenn es fehlerfrei assembliert wurde. Für all das können keine Kochrezepte gegeben werden – hier müssen die Kombinationsfähigkeit und Erfahrung eingesetzt werden.

Ein Tip ist, den ersten Pass immer auf den Drucker zu schicken, als Referenz, und den zweiten Pass auf Diskette abzulegen. Zwar kann der Quellcode eines Acht-KByte-Programms bis zu über 1000 Sektoren belegen, doch kann der Output so reduziert werden (Spalten 3 bis 6 sind meist überflüssig), daß der Umfang auf etwa ein Drittel zurückgeht. Aber eben erst hinterher. Man sollte immer einige formatierte Disketten bereithalten, besonders dann, wenn man nur über das nicht mehr zeitgemäße 90/180-KByte-Format des TI-Diskcontrollers mit DOS-40 verfügt.

Zudem ist es unbedingt empfehlenswert, zwei Laufwerke zu besitzen. Ist nur eines vorhanden, so muß sich ein eventuelles Ausgabefile auf der gleichen Diskette wie das zu untersuchende Programm befinden, was einen Diskettenwechsel dann, wenn ein DIS/FIX 80-File untersucht wird, unmöglich macht.

Bezugsquelle

DISKASSEMBLER kann, wie alle MG-Programme, bei LCC, Legio Computer Centre, Haagweg 169, 2281 AJ Rijswijk, Niederlande, bestellt werden. Telefon 070-995757.

Schlußwort

Diese Reihe von drei Softwarevorstellungen hatte verschiedene Ziele. Ein Hauptziel war sicherlich, diese außergewöhnlichen Programme einer größeren Zahl von TI-Anwendern in angemessener Form vorzustellen, auch um zu zeigen, daß es für den TI Software gibt, die man gemeinhin dem PC-Bereich zuordnen könnte. Ebenso sollte jeweils eine Art deutsches Handbuch entstehen, das die vielfältigen Probleme, die beim Lesen der englischsprachigen Manuals auftreten können, wenigstens teilweise beseitigen hilft. Zudem war es ein Experiment, das zeigen sollte, welche Resonanz eine sehr ausführliche Softwarebesprechung ohne jeden grafischen Schnick-Schnack erhält. Doch in der Hauptsache sollte diese Reihe zeigen, daß in dem TI noch jede Menge ungenutzter Möglichkeiten stecken.

Natürlich können diese drei Beiträge kein Handbuch ersetzen – so war es auch nicht geplant. Aber wenn sie dem einen oder anderen einen Zugang zu DIAGS, EXPLORER oder DISKASSEMBLER verschafften, so waren sie nicht umsonst.

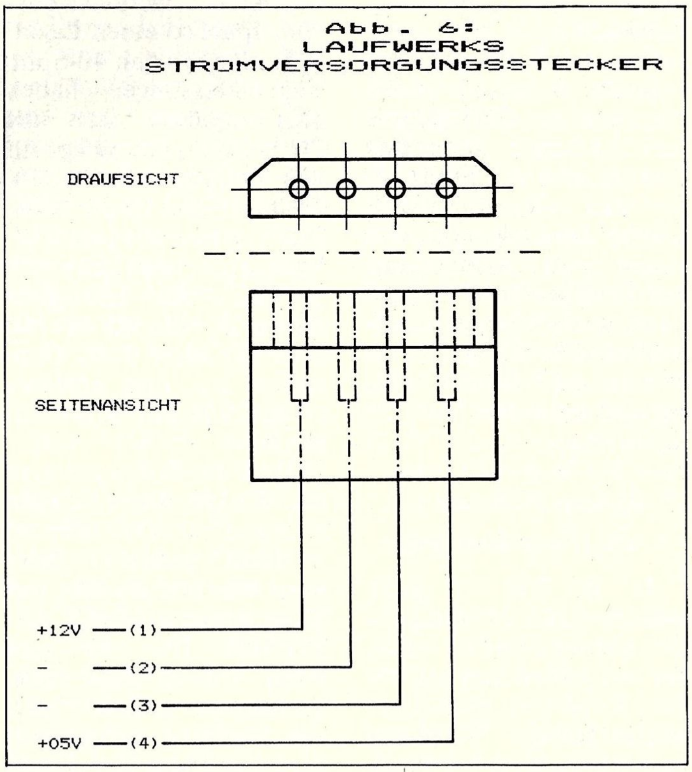
Christopher Winter
system-99-User-Group ■

All Systems Go, Teil 3

In der vergangenen Ausgabe hatten wir gerade bis zu vier Laufwerke an den Bus angeschlossen. Man spricht bei dieser elektrischen Anordnung der Laufwerke am SHUGART-BUS (also alle Drives am gleichen Kabelausgang des Controllers, selektiert über die Leitungen Drive Select 1 bis 4) von einer sog. DAISY-CHAIN-CONNECTION. Im Gegensatz dazu gibt es auch Controller, die für jedes Laufwerk einen eigenen kompletten Busausgang haben. Man spricht von einer RADIAL-CHAIN-CONNECTION. Hier gibt es eigentlich keinen wirklichen Unterschied im Betrieb. Genaugenommen haben alle Laufwerke nur einen echten Drive-Select-Eingang! Über eine Brücke auf der Laufwerksplatine wird nun festgelegt, welches Kabel des Busses an die Laufwerkselektronik gelangt. Man „programmiert" das Laufwerk als Drive 1,2,3 oder 4, indem man die Brücke einsetzt, die die Busleitung Drive Select 1, 2, 3 oder 4 mit dem internen Eingang verbindet. Auf Grund der Busgeometrie ist daher bei der Daisy-Chain-Connection das selektierte Ansprechen auf vier Laufwerke begrenzt.

Radial-Chain-Connection

Für die Anwendung der Radial-Chain-Connection haben die Laufwerke außer den Brücken 1 bis 4 noch eine fünfte. Diese Alternative, meist mit MX (o.ä.) bezeichnet, selektiert das Laufwerk unabhängig von den Drive-Select Eingängen (der „interne" Drive-Select-Eingang des Laufwerks wird dabei einfach fest auf



Massepotential gelegt und nicht über einen externen Transistor angesteuert.). Hier muß der Controller entscheiden, welchen kompletten Bus er bedient. Er hat ja, wie schon erwähnt, für jedes Laufwerk einen kompletten Busausgang. Die Selektion wird also hier im Controller vorgenommen. Auch muß bei der Radial-Chain-Connection der Terminatorwiderstandschip bei jedem Laufwerk eingesetzt werden. Die für den TI erhältlichen Controller arbeiten aber ausnahmslos mit der Daisy-Chain-Connection und max. 4 Laufwerken. Die Laufwerke, die TI bei der Controllerkonzeption zugrunde gelegt hat, hatten aber nur drei Drive-Select-Eingänge (1, 2 und 3). Dies ist daher die max. Ausbaustufe des TI-Kontrollers, der auf diese Laufwerke zugeschnitten ist. Neuere haben dagegen alle 4 Select Eingänge/Brücken. Will man ein „altes" Laufwerk an einen neuen Controller unter Nummer 4 als DSK4 ansprechen, braucht man nur das Bussignal Drive Select 4 mit dem internen Selektionseingang des Laufwerks zu verbinden. Aus den bisherigen Ausführungen ergibt sich, daß man diese Leitung nur mit dem gemeinsamen Pol der 3 Selektionsbrükken verbinden muß.

Bisher wurde noch kein Wort über die Stromversorgung der Laufwerkstypen und -formate verloren. Alle arbeiten, bis auf wenige hochmoderne Varianten, mit zwei positiven Versorgungsspannungen, +5V und +12V gegen Masse. Auch hier hat jede Spannung ihre eigene Masseleitung, wodurch sich die Steckverbindung der Stromversorgung als 4polig erweist. Bei nahezu allen 3" und 5.25" wird dafür ein Flachstecker verwendet, der verpolungssicher konstruiert ist. Seine Belegung ist verbindlich (siehe Abb. 6). Immer, wenn dieser Stecker Verwendung findet, ist er so belegt.

Alte und neue Laufwerkstypen unterscheiden sich nie in der Versorgungsspannung, sondern, wenn überhaupt, nur in der Stromaufnahme. Sie liegt im Betrieb zwischen etwa 100mA und 1A (Grenzwerte). Die tatsächliche Stromaufnahme ist natürlich vom jeweiligen Betriebszustand abhängig. So etwa davon, ob der Motor gerade läuft, der Schreib-/Lesekopfträger bewegt wird, das Drive im Schreib- oder Lesezustand ist, und so weiter. Wichtig ist nur, daß die Spannungsquelle (hier: das Netzgerät) einen genügend kleinen Innenwiderstand besitzt, um die erforderlichen Ströme zu jedem Zeitpunkt liefern zu können. Auch sollten die Toleranzen der Spannungsquelle 5% nach oben oder unten nicht überschreiten. Dies ergibt reale Werte von 4.75 bis 5.25 für die 5 Volt-Spannungsquelle und 11.4 bis 12.6 bei der 12-Volt-Quelle.

Die richtige Spannungs-Quelle

Diese Quellen müssen, bezogen auf den Strom, so „steif" sein, daß sich die Spannungen auch beim Einschalten der Mo-

toren usw., also dann, wenn die Stromaufnahme kurzzeitig deutlich über den Nennwerten liegt, nicht nennenswert ändern. Insbesondere wenn die Laufwerke ausschließlich aus der P-Box versorgt werden, sollte man hier lieber einmal mehr mit dem Voltmeter Klarheit schaffen. Achten Sie darauf, daß Ihnen die Spannungen nicht in die Knie gehen, wenn Sie sie entsprechend belasten, da die Laufwerke sonst nicht mehr wissen, was los ist und ihren Orientierungssinn verlieren. Werden die Spannungen über andere mechanische Steckverbindungen, als oben beschrieben, zugeführt, setzt man Adapter ein oder wechselt den Stecker am Kabel. Die Belegung ist aber gleich.

Wenn Sie das bis hierher Gesagte aufmerksam studiert haben, steht dem Einsatz von HuK-Laufwerken (Hinz und Kunz) nichts mehr im Wege. Für die Mehrzahl der User dürften sich damit alle quälenden Fragen beantwortet haben.

Abschließend wird aber noch ein Bonbon beschrieben, welches bei Insidern auf Interesse stoßen dürfte:

Leeren Sie an dieser Stelle bitte die Aschenbecher, kochen Sie frischen Kaffee (und stellen Sie ihn in Reichweite), lüften Sie den Raum und los geht's:

Bonbon für Insider

Wie schon gesagt, ist das Standardformat der Disketten am TI einfache Dichte, einseitig und 40 SPUR. Dies sind rund 90 KByte Kapazität (bei dem vom Kontroller intern verwendeten Format für die Spur- und Sektoreninitialisierung). Dieses Format läßt sich nur aufwendig ändern, so sinnvoll ist dies auch nicht! Will man mehr Daten auf die Laufwerke bringen, geht man wie eingangs beschrieben vor. Man verdoppelt die Dichte, ... Natürlich kann man auch zuerst die Kopfzahl verdoppeln. Jedenfalls endet der Spaß bei 360 KByte (bei 2 Köpfen, doppelter Dichte und 40 Spuren). So sind also 4 x 360 KByte das Optimum, wie einem jeder sagt. Aber warum eigentlich? Es gibt doch auch 80-Spur-Laufwerke! Die Dinger werden ja auch immer billiger und was wäre denn, ...

40 oder 80 Spuren

Ja, ja. Wie wir jetzt wissen, hat ein 40-Spur-Drive einen Schrittmotor, der bei jedem Stepimpuls den Kopf um eine Spur weiter bewegt. Wenn ich nun die Schrittweite halbiere, kann ich auch den Bereich zwischen zwei Spuren anfahren. Das ist aber dann ja schon ein 80-Spur-Drive. Schließe ich ein solches an den Controller an, kann ich natürlich meine Disketten nicht mehr lesen, weil nur Spur 00 bei beiden Formaten gleich ist. Spur 1 beim 80er ist aber Spur „0.5" beim 40er, usw. Wenn ich mit einer kleinen Schaltung zur Impulsverdoppelung nun aber jeden vom Controller kommenden Impuls verdopple und zum Laufwerk leite, habe ich aus einem 80-Spur ein 40-Spur-Drive gemacht. Die Spuren, die das 80er Drive mehr hat, werden so übersprungen. Die Industrie weiß dies auch. Und um die Kosten für die doppelte Lagerhaltung zu sparen, geht man vielfach dazu über und baut nur noch 80-Spur-Laufwerke mit integrierter Impulsverdopplerschaltung. Über eine weitere Brücke kann man dann wählen, ob das Drive als 40er oder 80er eingesetzt werden soll. Man spart so einen Lagertyp, nämlich den 40-Spur-Typ. Messungen haben aber ergeben, daß die Controller keineswegs an den 40 Spuren festhalten. So intelligent ist die DSR-Routine nun wieder nicht. Vielmehr bestimmt die Software beim Initialisieren und befiehlt die 40 Spuren. Mit dem Controller hat dies sowenig zu tun wie mit den Laufwerken.

Und die Software, also die Diskmanager, enden nun mal bei 40 Spuren.

Haste gedacht! Erinnern wir uns an den von TI gelieferten –DISKMANAGER II–. Er gibt beim Initialisieren zwei Stellen (4 und 0 = 40) vor. Gedacht war dies ursprünglich, um auch Uraltlaufwerke mit 35 Spuren (die gab's auch mal. Sie hatten die gleiche Schrittweite von Spur zu Spur, wie die 40er!) bedienen zu können. Man überschreibt die 40 dann einfach mit 35. Was glauben Sie, was der Controller macht, wenn Sie an den Step-Ausgang einen Ereigniszähler anschließen und im Diskmanager II 40 anstatt 99 eingeben? Richtig! Er gibt 99 Impulse aus. Dies ist natürlich sinnlos. Aber wenn Sie 80 eingeben ...

Dann bedient der Controller 80 Spuren. Wenn das Laufwerk also 80 Spuren hat, werden diese auch „versorgt"! Um das „alte" 40er-Format lesen zu können, ändern Sie die Brücke am Laufwerk auf 40 Spuren. Sie erreichen so jede 2te Spur, 40mal genau. Zweckmäßigerweise legen Sie die Umschaltung mit einem Schalter an die Gerätefront. Oder Sie ersinnen ein System, das den Disketten das Format ansieht (etwa ein reflektierender Aufkleber, der mit einer Fotozelle überwacht wird.).

Dieser Zusatz schaltet dann ein Relais, das seinerseits die Spurumschaltung vornimmt. Allerdings will ich ein Mißverständnis diesbezüglich verhindern: Bei 80 Spuren und 2 Seiten muß der Controller natürlich viel später von Seite 1 auf 2 umschalten, da sich die Zahl der Sektoren pro Seite vergrößert hat. Diese Angabe „steckt" in der Kontroller-DSR, die Sie ändern müssen, wenn Sie echten 80-Spur-Betrieb fahren wollen. Das dahingehend abgeänderte Routinchen, in Form eines EPROMs, wird unter dem bezugsreichen Namen DOS 80 vertrieben. Meines Wissens reicht diese Erweiterung aber nur bis 80 Spuren pro Seite, zweiseitig und einfache Dichte, womit Sie dann im 80er Modus auch nur 1440 Sektoren pro Diskette erhalten. Dieser Umfang kommt dann dem 40er-Modi DS/DD gleich. Aber ein bißchen Spielen, zum besseren Verständnis, sei erlaubt. Machen Sie sich doch mal den Spaß:

Leerdiskette einlegen, Diskmanager II einlegen und INIT aufrufen, 80 Spuren eingeben, einseitig und einfach dicht eingeben und laufenlassen. Vorher schaffen Sie noch die Möglichkeit, den STEP-Impuls am SHUGART-BUS während des Betriebs der Laufwerke zu unterbrechen. Dies ist aber nur notwendig, wenn Sie nicht sicher sind, daß der Stepmotor im Drive nicht durch An-

steuerung über Spur 39 hinaus dejustiert wird. Lassen Sie nun den Diskmanager „laufen“ und unterbrechen Sie ggf. den Stepimpuls ab Sektor 350dez und schalten Sie ihn wieder zu, wenn der Manager zum Lesen ansetzt. Kommt die Routine dann beim Lesen bei Spur 35 an, schalten Sie den Rechner nach der Anzeige von Sektor 350 oder 351 kurzerhand aus. Wenn Sie dann wieder einschalten und den Katalog anwählen, werden Ihnen 360 anstatt 720 Sektoren angeboten. Funktionieren kann es natürlich nur bis 358 oder so, weil der Rest physikalisch gar nicht existiert. Wenn man jetzt 40/80-Spur-Laufwerke hätte, würden einem bei 80 Spur, doppelseitig und doppeldicht sage und schreibe 2880 (kein Druckfehler) Sektoren winken. Das sind dann lockere 1 MByte unformatiert/720 KByte formatiert und somit bei 4 Drives 4 MByte unformatiert. Der Grund, warum man in der Praxis nicht so unmittelbar 2880 Sektoren erhält, ist oben ja gesagt worden. Es sollte aber nun einleuchten, daß nichts an den Laufwerken oder dem Controller bzw. dem Bus zu ändern ist.

Der 64er-Freund mit seinem Satz: „Mein 64er-System ist aber was ganz anderes“, hat wohl recht. Aber ein 300 SEL 6.3 ist nun mal kein Golf-Diesel.

Bevor ich zum Ausdrucken dieses Artikels ein neues Farbband erstehe, noch ein paar Tips, wenn Ihre Laufwerke nicht mehr richtig arbeiten, aber kein offensichtlicher Fehler zu erkennen ist:

Reinigen Sie die Fotobauteile (Diode und Transistor, o.ä.), die das Indexloch auf der Diskette abtasten und daraus den Indeximpuls generieren, mit einem Staubpinsel! Je nach Laufwerk trifft man auch hin und wieder auf eine Schlitzmaske aus Kunststoff, die sich im Strahlengang befindet, um Fremdlichteinwirkungen auszuschließen. Ein „verschliffener“ Indeximpuls wird vom Controller durchaus als nicht vorhandene Diskette und damit über die Software als Diskerror 16, usw. verstanden, da, wie schon gesagt, die Index-Leitung die einzige ist, die dem Controller sagt, ob sich was tut.

Aus dem Text geht hervor, warum die Einhaltung der Solldrehzahl von 300Upm ein wichtiger Parameter für eine einwandfreie Funktion des Gesamtsystems, ja eine Grundvoraussetzung ist. Wenn sich die beim Einjustieren durch den Hersteller vorgegebene Drehzahl beispielsweise durch Bauteilealterung und damit Werteänderung verstellt, kommt es zu Fehlfunktionen, die man in ihrer Vielfalt hier gar nicht alle beschreiben kann. Je nach dem, wie stark die Verstellung/Abweichung ist, kann der Controller bestimmte Sektoren nicht mehr lesen/beschreiben, findet den Katalog nicht mehr, eben alles, was mit der Drehzahl in Zusammenhang steht. Mit genügend Ruhe und Geduld kann man dies leicht, auch ohne teueren Einmeßplatz für Laufwerke, beheben. Auf der Platine des Laufwerks findet man einen Trimmwiderstand, der die Solldrehzahleinstellung ermöglicht. Legen Sie einfach eine, mit vielen kleinen Dateien beschriebene volle Diskette in den fehlerhaften Antrieb ein und versuchen Sie ohne Unterbrechung mit dem Diskmanager den Katalog zu lesen. Verstellen Sie dabei den Trimmer mit viel Gefühl so lange, bis der Katalog vollständig erscheint. Wenn alles einwandfrei eingelesen wird, versuchen Sie im Anschluß den Zugriff auf alle Dateien. Gelingt dies nicht, beginnen Sie noch einmal von vorn. Für den Fall der Fälle sollte man eine solche Diskette, oder besser ein paar davon, immer bereithalten, denn wenn Sie nur ein Laufwerk haben, können Sie im Fehlerfall ja keine zum Testen erstellen! In der Regel ist auf dem Motor auch eine sogenannte Stroboskopscheibe mit

Stroboskop-Effekt

Hz angebracht. Wenn Sie diese Scheibe beim Betrieb des Laufwerks mit Hz-Lichtimpulsen „beleuchten“, kommt die entsprechende Teilung bei richtiger Drehzahl des Motors zum Stehen (vergleiche auch –Stroboskopischer Effekt–). Eine gute Möglichkeit, die Laufwerksdrehzahl von Zeit zu Zeit einfach zu überprüfen. Desweiteren kann man auch an die Index-Leitung einen Frequenzzähler, falls vorhanden, anschließen. Die wohl eleganteste Lösung. Sie erinnern sich: 300 Upm – 5 Ups – 5 Impulse/Sekunde (softwaresektoriert) – 5 Hz Drehfrequenz – 0.2 Sekunde Periodendauer.

Ein weiterer beliebter Fehler ist die Tatsache, daß sich die Nullmarke der Spuren verschiebt. Sie wissen ja jetzt, daß sich, wenn der Schreib-/Lesekopf über Spur 00 ist, über die Signalleitung –Track 00– dies dem Controller mitgeteilt wird. Ob der Kopf dann tatsächlich über Spur 00 ist, kann man davon aber nur bei einem einwandfrei arbeitenden Laufwerk ableiten. Verschiebt sich der mechanische Anschlag im Laufwerk, weil vielleicht die Schrauben sich durch Vibration auf Dauer gelöst haben, steht der Kopf eben nicht mehr richtig über der ersten Spur, und da alle anderen davon auch betroffen sind, tut sich nichts mehr. Auf den auf der ersten Spur untergebrachten Katalog der Diskette kann man so schon gar nicht mehr zugreifen. Versuchen Sie auch hier permanent mit dem Manager die Diskette zu katalogisieren und manipulieren Sie mit dem Finger den Anschlag. Kommt plötzlich an Stelle der Fehlermeldung der Katalog, müssen Sie nur noch den Anschlag neu justieren.

Gerne würde ich hier genauer werden, aber auf Grund der unüberschaubaren Vielfalt der firmeninternen Realisation der Laufwerke hat dies nur in Serviceunterlagen Sinn. Grundsätzlich können Sie mit den Drives mehr anstellen, als Sie glauben.

Mit Vorsicht behandeln...

Respekt ist ausschließlich bei der Behandlung des Schreib-/Lesekopfes und des Trimmers für die Einstellung der sog. Jitterfreiheit auf der Laufwerksplatine angebracht. Behandeln Sie den Kopf mit angebrachter Vorsicht und verwechseln Sie den Jittertrimmer nicht mit dem für die Drehzahl. Befassen Sie sich auch ruhig mal vorbeugend mit dem „Innern“ Ihrer Laufwerke. Dann blicken Sie, wenn's mal drauf ankommt, besser durch.

Am theoretischen Ende dieser kleinen Serie noch ein paar Richtigstellungen, die unnötige Leserbriefe ersparen sollen. In Teil 1 der vorletzten Ausgabe der HCA konnte unter Umständen der Eindruck erweckt werden, daß der beschriebene Sektorenzugriff so unmittelbar für den TI gilt. Dem ist nicht so. Es ist auch keine Beschreibung eines Zugriffs auf eine hardsektorierte Diskette, wie ein Leser behauptet hat, da nur von einem Indexloch ausgegangen wird, und es sich somit zweifelsfrei um eine softsektorierte Diskette handeln muß. Das Beispiel selbst sollte nur dem Laien darlegen, wie über die Bussignale gearbeitet werden kann bzw. wieviele Varianten möglich sind. Bei differenzierter Betrachtung (hier mal nicht nach dt) am TI selbst sei gesagt, daß die Anlage nach den Adressen auf der Diskette sucht, die beim Initialisieren auf diese geschrieben werden und die somit bei gleichem Initialisierungsformat bei allen Disketten gleich sind (von dubiosen Formaten zwecks Kopierschutz usw. einmal abgesehen!). Der Indeximpuls ist aber dennoch NICHT überflüssig, da die TI-DOS-Systeme das bereits beschriebene READY-Signal nicht vom Laufwerk erwarten, sondern unter Bezugnahme auf eben den INDEX-Impuls auf der Platine des Controllers generieren. Fehlt der Impuls, oder weicht er in seinen Parametern von den zu erwartenden ab, glaubt der TI Ihnen nicht, daß Sie eine Diskette im Drive haben und, folglich, springt die DSR nicht in ihre Lese- bzw. Schreibuntertitel, wodurch ein Datenhandling erst gar nicht zustande kommt.

Das erwähnte Beispiel ist aber meines Erachtens nicht das schlechteste, wenn es darum geht, dem Outsider das Prinzip etwas näher zu bringen. Darüber hinaus soll dieser Artikel dem Laien die technische Einbindung der Drives vermitteln und kein Referat über Adreßformate am TI-DOS sein. Aber dies ist natürlich eine von meinen Ansichten geprägte Meinung, die alle, die sich in der Lage glauben, komplexe technische Vorgänge, auf das im jeweiligen Fall Wesentliche reduziert, beschreiben zu können, aufruft, sich an die Arbeit zu machen und nicht durch Schweigen ihre mögliche, aber nicht bewiesene Qualifikation zu einer rein akademischen Tatsache verkümmern zu lassen und das einzige, was sich daraus ergibt, ein unzumutbarer Dünkel ist.

Merke: Im Stillen verfaßte Kritik ist immer leichter, als es in einer lauten Umgebung besser zu machen! *Jakob Brilmayer*

Fortsetzung folgt

Nur HCA bietet kostenlose Privat-Anzeigen

Dialog

Auch in dieser HCA wollen wir wieder eine Leseranfrage beantworten, die sicher für viele unserer Leser von Interesse sein dürfte. Die Fragen stammen von Johann Brescher, beantwortet werden sie von Christopher Winter von der System-99 User Group.

1. Frage:
Ist es möglich, zwei Konsolen gleichzeitig an die Box anzuschließen und damit von zwei Standorten auf die selben Daten Zugriff zu haben?

Diese Frage kann unter gewissen Voraussetzungen mit ‚Ja' beantwortet werden. Voraussetzung 1 ist, daß beide Konsolen eine eigene, vorzugsweise eingebaute 32K-Speichererweiterung besitzen und daß durch die eingesetzte Software sichergestellt ist, daß nicht gleichzeitig von beiden Systemen in die gleiche Datei geschrieben wird. Genaugenommen dürfen Sie nicht einmal in verschiedene Dateien einer Diskette schreiben, sofern Sie beide gleichzeitig offen halten. Die weiteren Details können hier jedoch nicht näher ausgeführt werden. Bei reinen Input-Files dürften jedoch keine Probleme auftreten.

Die zweite Voraussetzung ist weiterhin, daß Sie KEINEN Original TI-Disk-Controller in der P-Box haben.

Da Ihr System nun über zwei getrennte VDP-RAM-Einheiten verfügt, werden sowohl die Informationen über zuletzt benutztes Laufwerk als auch die jedem der drei Laufwerke zugeordnete aktuelle Spur fehlerhaft übergeben. Wollen Sie trotzdem mit einem TI-Controller arbeiten, so müssen Sie über ein permanent aktives Hintergrundprogramm diese 4 Byte zwischen beiden VDP-RAM-Einheiten austauschen, d.h. identisch halten. Alle Controller von Fremdfirmen heben diese Informationen in einem RAM auf der Controllerkarte auf, wodurch dieses Problem nicht auftaucht.

An Hardware benötigen Sie ein Flex-Cable-Interface, das Sie mit 2 Eingängen ausrüsten, oder ein zweites Interface (schaltungstechnisch aufwendiger). Da sich die beiden Konsolen nun Diskcontroller, Schnittstelle und evtl. Horizon RAM-Disk (mit anderen RAM-Disks geht es nicht wegen der separaten 32K) teilen sollen, müssen Sie Ihrerseits per Hardware sicherstellen, daß immer nur ein Rechner auf die Peripheriegeräte zugreifen kann. Ggfs. muß ein zeitgleicher Zugriff über READY synchronisiert werden. Sie müssen mit einer solchen Hardware nur die Aktivierung der vorhandenen DSR-Seiten überwachen und dann die entsprechenden Daten- und Adreßbustreiber des jeweiligen Flex-Cable-Interface zur entsprechenden Konsole durchschalten. Ihre Rechnerzuweisung sollten Sie über den Adreßtakt Phi 3 synchronisieren.

2. Frage:
Außer dem TI-99 besitze ich noch einen Taschencomputer TI-74 BASICALC. Das BASIC der beiden ist unterschiedlich. Ist es trotzdem möglich,

Daten (z.B. über den Kassettenrecorder) zwischen den beiden auszutauschen?

Um Daten über Dassette auszutauschen, muß das Datenformat übereinstimmen oder wenigstens bekannt sein. Hier können wir Sie nur an Texas Instruments verweisen. Mit dieser Kenntnis wäre es dann zumindest denkbar, eine Leseroutine für den TI-99/4A zu schreiben, ggfs. mit zusätzlichem Hardwareaufwand, je nach Aufzeichnungsverfahren des TI-74. Doch muß Ihnen in einem anderen Punkt widersprochen werden: Das im Basicalc eingebaute BASIC ist dem TI-Extended BASIC nicht nur ähnlich, beide sind sogar fast identisch. Hier sollte es ein Leichtes sein, eine direkte Umsetzung zu erreichen (durch Abtippen).

In der HCA 6/88 wurden einige Fragen von Herrn Whybrew beantwortet. Leider sind uns bei der Antwort zu Frage 5 einige sachliche Fehler unterlaufen, die wir so nicht stehenlassen wollen. Und um keine Unklarheiten entstehen zu lassen, wird diese Frage hier noch mal abgedruckt und korrekt beantwortet.

Frage:
Was genau ist ‚Genial Traveller', was kostet er und wo kann man in den USA hinschreiben, um ihn zu bekommen? DM 20,– pro Diskette scheint mir zu teuer zu sein.

TRAVelER (schreibt sich wirklich so!) in wenigen Worten zu beschreiben, ist fast unmöglich. Daher zur Klärung hier nur so viel:

Der TRAVelER geht auf eine Idee von Barry Traver zurück (daher die etwas eigenartige Großschreibung), nach der sich einige amerikanische Programmierer zusammengetan haben, um eine Art Diskettenzeitschrift (Diskazine) aufzuziehen. So vertreiben sie Programme wie Archiver, Systex etc. auf Disketten mit diesem Namen. Als Editor zeichnet Barry Traver verantwortlich; vertrieben werden die Disketten (bisher immer Flippies, also im Prinzip je 2 Disketten) von Genial Computerware.

Eine Kontaktadresse wäre z.B. die Chicago TI-User Group, zu deren Autoren Barry Traver zählt oder aber Barry Traver selbst, in 835 Green Valley Drive, Philadelphia, PA 19128. Ein Rückumschlag (adressiert und frankiert oder eine IRC, International Reply Card) wird erwartet.

TRAVelER ist de facto NICHT Public Domain oder Freeware, da sich die Autoren das Copyright in der Regel vorbehalten, mit Ausnahme von beigelegten Freeware-Programmen sowie der in letzter Zeit beigelegten Bonus Disks des Sohnes von BAT (so die USA-übliche Abkürzung für B.A. Traver).

Die Ihnen für DM 20,– angebotenen Disketten sind Raubkopien, da TRAVelER nach Maßgabe von BAT NICHT kopiert werden darf. Er fordert in jeder Ausgabe dazu auf, keine Kopien weiterzugeben und schlägt vor, den etwaigen Interessenten stattdessen ein Abonnement zu empfehlen. ■

Club-Ecke

In der Chemischen Verbindung 77 e.V., einem naturwissenschaftlichen Verein in Duisburg, kristallisierte sich die Informatik-Gruppe als die aktivste und mitgliederanziehendste heraus. Für die Zusammenarbeit mit einem Duisburger Anzeigenblatt, dem StadtPanorama, entschloß man 1984 die Auskopplung der Informatik-Gruppe als eigenständigen Club PCC, den Panorame Computer Club. Dieser war von Anfang an für alle Computertypen offen, man sah aber sehr schnell ein, daß ein sinnvolles Arbeiten nur in Computer-spezifischen Arbeitsgruppen möglich sein werde, so war die erste Arbeitsgruppe der TI-99/4A SERVICE.

Seit 1985 beläuft sich die Zahl der Mitglieder auf 20, wobei Mitglieder ohne Komplettanlage diese entweder ausbauten oder durch solche mit Box ausgetauscht wurden. Auch wenn wir bisher nach außerhalb von Duisburg haben wenig von uns hören lassen, so sind wir mindestens so aktiv wie alle anderen TI-Clubs, zu denen wir natürlich gute Kontakte pflegen, im Bereich Hardware (z.B. Eisenbahn-Steuerung) und Software (z.B. Turbo-Pascal-Grafik-Erweiterung), es wäre wohl müßig, hier alles im einzelnen aufzuzählen.

TI-99/4A SERVICE
c/o Hans Greiffenberg
Großglocknerstraße 45
4100 Duisburg 28
Tel. 0203/724475

INITIATOR:
Franz Markus Kraemer
Trarbacher Str. 15 a
4100 Duisburg 25
Tel. 0203/783833

Geschäftsstelle:
Am Thelenbusch 123
4100 Duisburg 29
Tel. 0203/763920

Wer Interesse hat, sollte auf jeden Fall mal auf ein Treffen kommen, es findet alle 14 Tage im STUDIO DER HEIMAT, Am See 22, in Duisburg-Wedau (nähe Sechs-Seen-Platte) statt, oder sich bei der Kontaktadresse melden. (Es werden doch mehr als 20 TI-User im Ruhrgebiet herumhacken, oder?).

Vorankündigung

Wir, der PCC aus Duisburg (der vielleicht schon eher in der TI-Revue von sich hätte hören lassen sllen), sind es, die das 3. Internationale TI-99/4A USER-Treffen in Duisburg organisieren.

Dieses Treffen wird Anfang Oktober, möglicherweise am 01.10.1988, stattfinden und wieder Neuestes und alt Bewährtes rund um den TI-99/4A, und zwar nur um diesen, zeigen.

Über 30 (in Worten dreißig) Clubs aus dem In- und Ausland sind eingeladen, ebenso Händler und selbstverständlich die TI-Revue. Eine Mailbox enthält ebenfalls Infos über das Treffen unter den Punkten Kultur, Termine u.a., die Daten sind: MEGALON-BOX, Tel. 02151/40161, 300 Baud, 8 Bit, 1 Stop Bit, No Parity. Voraussichtlich wird diese Box auch ein spezielles TI-Brett bekommen. Außerdem möchten wir schon an dieser Stelle darauf hinweisen, daß wir eine Gebrauchtwaren-Börse veranstalten. ■

```
100 ! **********************
110 ! *                    *
120 ! * XLOADCS POKELISTE *
130 ! *                    *
140 ! *     M. EICHHORN    *
150 ! *  GOETHESTRASSE 7   *
160 ! * 6242 KRONBERG/TS.  *
170 ! *                    *
180 ! **********************

190 !
200 CALL INIT
210 ! Programm-Datas laden
220 FOR I=0 TO 50
230 READ A(1),A(2),A(3),A(4)
,A(5),A(6),A(7),A(8)
240 CALL LOAD(9460+I*8,A(1),
A(2),A(3),A(4),A(5),A(6),A(7
),A(8))
250 NEXT I
260 ! Def-Table setzen
270 CALL LOAD(16376,76,79,65
,68,67,83,37,36)
280 ! Pointer auf Def-Table
285 CALL LOAD(8196,63,248)
290 ! Start des Programms
295 CALL LINK("LOADCS")
300 END
310 DATA 0,0,0,0,0,0,0,0,0,0
,0,0,203,20,203,53
320 DATA 203,78,203,231,204,
71,204,150,204,228,205,29,20
5,75,205,96
330 DATA 5,0,16,0,0,0,33,0,9
6,3,67,83,49,0,8,0
340 DATA 2,224,36,244,4,195,
2,0,15,0,2,1,37,20,2,2
350 DATA 0,13,4,32,32,36,2,0
,131,74,204,32,37,30,212,32
360 DATA 37,32,2,10,15,13,20
0,10,131,86,2,0,0,3,200,0
370 DATA 131,84,4,224,131,20
8,216,32,37,34,131,109,216,3
2,37,33
380 DATA 131,124,4,32,38,46,
0,61,2,0,16,0,2,1,37,0
390 DATA 2,2,0,6,4,32,32,44,
2,0,16,6,2,131,0,0
400 DATA 22,1,193,8,5,131,19
2,72,2,129,39,0,26,6,192,135
410 DATA 4,32,32,44,193,134,
19,17,16,198,2,0,15,0,200,0
420 DATA 131,28,2,0,11,0,4,3
2,32,52,0,0,1,224,2,0
430 DATA 3,14,4,1,5,6,6,0,7,
245,2,1,37,174,2,2
440 DATA 0,8,192,49,4,32,32,
48,6,2,22,251,2,0,3,0
```

Die Quelle sprudelt weiter

In der HCA 3/88 wurde der Quellcode eines Programmes zum Laden von PROGRAM-Files mit dem Extended-BASIC-Modul, der 32KByte-Erweiterung und dem Disk-System abgedruckt. Da aber nicht alle TI-User ein Diskettenlaufwerk besitzen, bietet das folgende Programm die Möglichkeit, diese PROGRAMM-Files von Kassette zu laden. Ein weiteres Programm, mit welchem man Files im Memory-Image-Format von Diskette auf Kassette übertragen kann, erscheint in einer der nächsten Ausgaben der HCA.

Zum Format des Files nur soviel: Assembler-Objektcode-Files können mit Hilfe der SAVE-Routine, die mit dem Editor/Assembler-Modul geliefert wird, in das sogenannte Memory-Image-Format übertragen werden, welches den Vorteil besitzt, wenig Platz zu belegen und sehr schnell geladen zu werden. Dieses Format ist, wie der Name schon sagt, ein direktes Abbild des Speichers, Byte für

```
450 DATA 4,193,192,160,131,1
12,4,32,32,32,5,128,128,128,
18,251
460 DATA 2,0,9,0,200,0,131,7
4,216,32,37,33,131,124,4,32
470 DATA 38,46,0,24,2,0,11,0
,200,0,131,74,216,32,37,33
480 DATA 131,124,4,32,38,46,
0,74,4,84,4,224,38,66,4,96
490 DATA 48,124,4,192,2,8,22
4,129,216,8,131,212,2,8,160,
129
500 DATA 216,8,140,2,6,200,2
16,8,140,2,38,14,38,52,4,224
510 DATA 200,62,131,236,216,
32,152,2,38,50,216,32,152,2,
38,51
520 DATA 6,32,38,50,2,224,13
1,224,2,13,152,0,209,32,131,
115
530 DATA 9,132,2,36,131,2,2,
5,38,110,205,5,2,5,0,58
540 DATA 197,5,6,196,216,4,1
31,115,4,96,0,96,6,96,131,11
5
550 DATA 2,224,38,14,216,32,
38,50,156,2,216,32,38,51,156
,2
560 DATA 3,128,0,0,0,0,0,0
```

Byte, wobei jeweils acht KByte unter einem Filenamen gespeichert werden können. Jedem File vorangestellt wird ein Header aus sechs Byte bzw. drei Wörtern. Das erste Wort zeigt an, ob das File allein lauffähig ist oder noch weitere Files in den Speicher geladen werden müssen. Das zweite Wort gibt die Länge des Files in Byte an (*ohne* die sechs Headerbyte), während das dritte Wort die Adresse angibt, ab der das File geladen wird.

Nun zum Laden und Starten des Programms: Nach dem Eintippen der Poke-Listen das Programm unbedingt abspeichern! Nach der Eingabe von RUN vergehen zirka 20 Sekunden, bis die Data geladen sind, danach erscheinen die bekannten Aufforderungen zum Starten des Kassettenrekorders.

Nach Laden des ersten Files überprüft das Programm, ob weitere Files zu laden sind und gibt gegebenenfalls erneut die Aufforderung zum Starten des Rekorders. Wenn das lezte File geladen ist, werden innerhalb von etwa zwei Sekunden das VDP-RAM initialisiert und die Zeichensätze geladen. Dies ist erforderlich, um auch solche Programme laden und starten zu können, deren Bildschirmausgabe speziell für das EDITOR/ASSEMBLER-Modul ausgelegt wurde (siehe HCA 3/88).

Danach startet das geladene Maschinenprogramm.

Mathias Eichhorn, TI-Workshop Wiesbaden■

```
*          ***********
*         *************
*         **         **
*         ** XLOADCS **
*         **         **
*         *************
*          ***********
*
*         (C) M. Eichhorn, Goethestr. 7, 6242 Kronberg/Ts.
*
*         Loader fuer PROGRAM-Files unter XBASIC
*         Benoetigte Konfiguration: XBASIC und 32K-RAM
*
*         Aufruf des Programms mit:
*
*         CALL LINK("LOADCS")
*
*
          AORG >24F4          Programm liegt im Low-Mem
*
          DEF  LOADCS
*
*
VSBW      EQU  >2020
VMBW      EQU  >2024
VSBR      EQU  >2028
VMBR      EQU  >202C
VWTR      EQU  >2030
ERR       EQU  >2034
*
MYWS      BSS  32
*
PABBUF    DATA >0500          Opcode fuer LOAD
          DATA >1000          VDP-Buffer ab >1000
          DATA >0000
          DATA >2100          max. >2100 Bytes laden
          DATA >6003          Screen-Offset und Namenlaenge
DEVICE    TEXT 'CS1'
*
B00       BYTE 0
B08       BYTE 8
*
          EVEN
*
LOADCS    LWPI MYWS           Eigenen Workspace laden
          CLR  R3
*
PABSET    LI   R0,>0F00       PAB in VDP schreiben
          LI   R1,PABBUF
          LI   R2,13
          BLWP @VMBW
          LI   R0,>834A
          MOV  @DEVICE,*R0+       Devicenamen in FAC
          MOVB @DEVICE+2,*R0
          LI   10,>0F0D           Pointer auf 1. Byte
          MOV  10,@>8356             nach Devicenamen
          LI   R0,>0003           Devicenamenlaenge
          MOV  R0,@>8354
          CLR  @>83D0             GROM-Searchpointer loeschen
          MOVB @B08,@>836D        Zeiger fuer DSRLNK
          MOVB @B00,@>837C        GPL-Statusbyte loeschen
          BLWP @GPLLNK            Kassettenroutine aufrufen
          DATA >003D
*
          LI   R0,>1000
          LI   R1,MYWS+12
          LI   R2,>0006
          BLWP @VMBR              Startparameter uebernehmen
          LI   R0,>1006                in R6, R7, R8
          CI   R3,0               Erstes File ?
          JNE  LOAD1
          MOV  R8,R4              Ja --> Startadresse sichern
LOAD1     INC  R3                 Zaehler fuer Files erhoehen
          MOV  R8,R1              R8 --> Startadresse
          CI   R1,>2700           Startadresse im Loader-Bereich?
          JL   ERROR              Ja --> Error
          MOV  R7,R2              R7 --> Byteanzahl
          BLWP @VMBR              File von VDP in RAM schreiben
          MOV  R6,R6              R6 = 0?
          JEQ  START              Ja --> Programm starten
          JMP  PABSET             Nein --> naechstes File laden
*
ERROR     LI   R0,>0F00           PAB-Adresse
          MOV  R0,@>831C
          LI   R0,>0B00           Fehlermeldung 'MEMORY FULL'
          BLWP @ERR
*
VDPREG    DATA >0000,>01E0,>0200,>030E
          DATA >0401,>0506,>0600,>07F5
*
START     LI   R1,VDPREG          VDP-Register setzen
          LI   R2,>0008
LOOP      MOV  *R1+,R0
          BLWP @VWTR
          DEC  R2
          JNE  LOOP
          LI   R0,>0300           VDP-RAM loeschen
          CLR  R1
          MOV  @>8370,R2          Hoechste Adresse im VDP-RAM
LOOP1     BLWP @VSBW
          INC  R0
          C    R0,R2
          JLE  LOOP1
*
          LI   R0,>0900           Zeichensaetze laden
          MOV  R0,@>834A
          MOVB @B00,@>837C
          BLWP @GPLLNK
          DATA >0018
          LI   R0,>0B00
          MOV  R0,@>834A
          MOVB @B00,@>837C
          BLWP @GPLLNK
          DATA >004A
          B    *R4                Start des Programms
*
SUBWS     BSS  32                 Subroutine-WS fuer GPLLNK
GPLLNK    DATA SUBWS,GPL1
GRMADR    BSS  2
*
GPL1      MOV  *R14+,@>83EC
          MOVB @>9802,@GRMADR
          MOVB @>9802,@GRMADR+1
          DEC  @GRMADR
          LWPI >83E0
          LI   R13,>9800
          MOVB @>8373,R4
          SRL  R4,8
          AI   R4,>8302
          LI   R5,GPLRT-2
          MOV  R5,*R4+
          LI   R5,>003A
          MOV  R5,*R4
          SWPB R4
          MOVB R4,@>8373
          B    @>0060
GPLRT     DECT @>8373
          LWPI SUBWS
          MOVB @GRMADR,@>9C02
          MOVB @GRMADR+1,@>9C02
          RTWP
*
          END
*
```

2.5 Hinweise zum Einbau der Zusatzplatine in die Konsole
****

In Abbildung 13 ist der Bestückungsplan der Leiterplatte dargestellt.

Bei der Bestückung und Inbetriebnahme geht man am besten so vor, daß zuerst alle Durchkontaktierungen eingesetzt werden, dann die Widerstände und die LS-Schaltkreise. Danach nimmt man diesen Teil erst einmal in Betrieb und testet die Funktion aller Logikkombinationen. Dazu bedarf es keiner besonderen Hilfsmittel.

Dann bestückt man den OPV-Teil und den '4093. Wenn auch dieser Schaltungsteil funktioniert, können die RAMs eingesetzt werden. Vorher sind die Leitungen unter den Schaltkreisen mit der Lupe exakt zu untersuchen (!), am besten werden sie verzinnt, auch die Lötaugen der RAMs sind vor dem Einlöten der Chips sauber zu verzinnen, damit das Einlöten problemlos funktioniert.

Die zusätzlichen Leitungen von der Leiterkarte zum indirekten Steckverbinder werden entsprechend dem Belegungsplan direkt verdrahtet.

Auf jeden Fall müssen die Pull-Up-Widerstände für die Eingänge/EXIO und GROMEN eingesetzt werden.

Die Leiterplatte wird mit der Bestückungsseite zur Konsolenplatine unter dieser angebracht. Die Positionierung ergibt sich automatisch, wenn man an alle Datenleitungen ca. 15 mm lange (am besten flexible) Leitungen anbringt. Diese Leitungen werden dann direkt an die Datenpins der CPU angelötet. Die Leitungen zu den Adreßpins der CPU sind etwa 20 mm länger zu halten. Das Layout der Zusatzplatte ist so gehalten, daß die Leiterkarte bei dieser Befestigungsart sich auch 90 Grad ausschwenken läßt (z.B. zu Testzwecken), wenn die entsprechenden Leitungen lang genug gehalten werden. Zuvor sollten aber alle notwendigen Unterbrechungen auf der Platine vorgenommen werden:
Bitte orientieren Sie sich dabei am Bestückungsplan des Konsolen-Prints (Abb. 14) sowie an den Abbildungen 5 bis 12!

a) Erzeugung der Signale BE, A, BA (Punkte A und B)

Der Ausgang des Gatters U 507.3 Pin 8 wird unmittelbar hinter dem Pin 8 auf der Leiterseite abgetrennt, —> Pin 8 wird das Signal BE (/8000 ... /83FF).

Leitung abtrennen

Abtrennen der Leitung des Pin 12 der IS U 606 auf der Bestückungsseite in Höhe des Pin 30 der CPU. Beachte, IS 606 und IS 603 sind im Bestückunsplan der TI-Unterlagen vertauscht eingezeichnet (!), IS U 606 befindet sich unter IS U 605, IS U 603 unter IS U 604. Pin 12 und U 606 erhält das Signal BA: (RAMEN v EXIOEN).

Dadurch bleibt der Leiterzug von /RAMEN dazwischen übrig, er erhält das Signal A (/RAMEN).

Die Abbildungen 5 und 6 unterstützen Sie bei der Ausführung der notwendigen Änderungen.

b) Erzeugung der Signale CE und CA (Punkt C)

Die Leitung U 613

Dem TI un
Hem

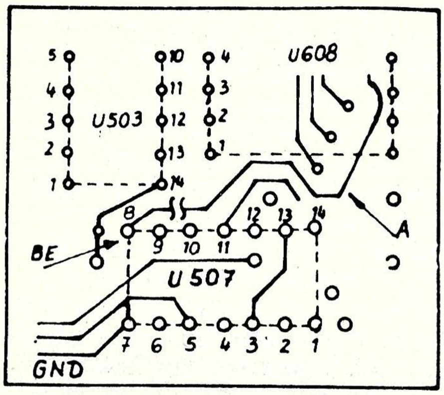


Abb. 5 Leiterseite Leitung von Pin U 507.8 abtrennen, der das Signal BE liefert. Der Rest der Leitung heist A (IRAMEN)

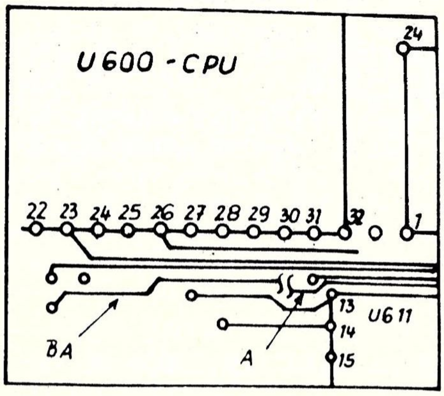


Abb. 6 Bestükkungsseite Leiterzug von IS 606.12 abtrennen, er erhält das Signal BA, der Rest heist A.

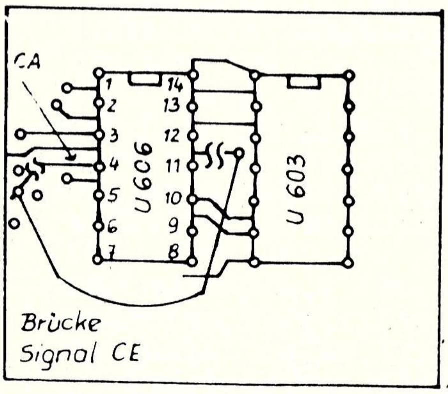


Abb. 7 Bestükkungsseite Pin U 606.4 aus der Leitung entfernen und überbrücken.

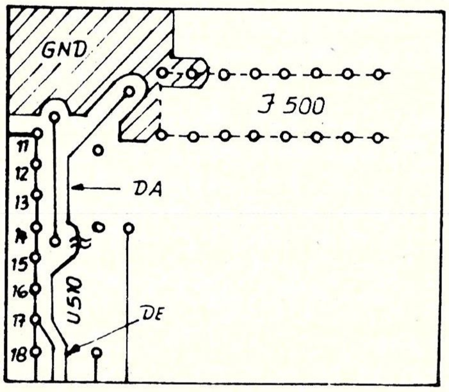


Abb. 8 Leiterseite Leitung von U 508.12 neben Durchkontaktierung trennen. Pin U 508.12 liefert dann das Signal DE, DK wird zum Signal DA.

ers
geschaut

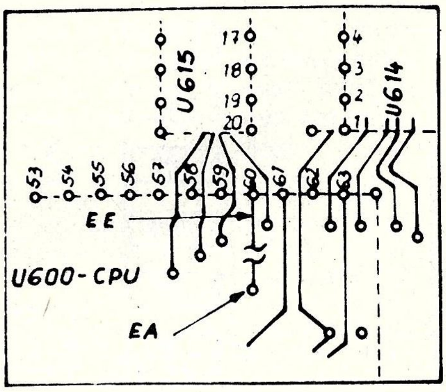


Abb. 9 Leiterseite CRUCLK von U 600.60 auftrennen, U 600.60 wird dadurch zum Signal EE, DK zum Signal EA.

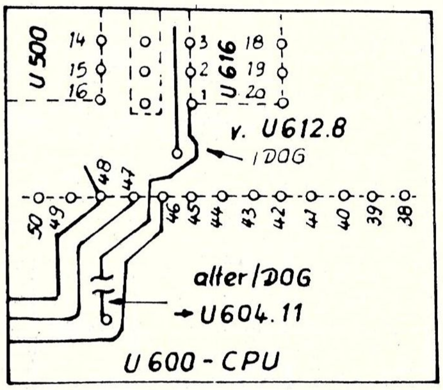


Abb. 10 Leiterseite
Altes /DOG entfernen (kommt von IS 603.8) und neues /DOG anlegen.

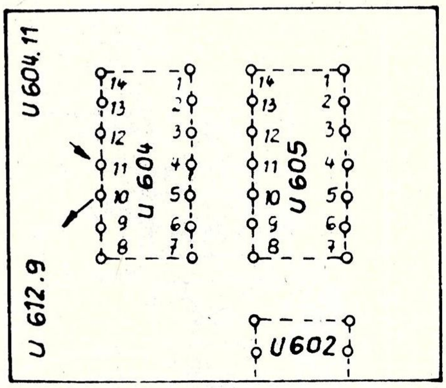


Abb. 11 Leiterseite
Ehemaliges /DOG am freien Inverter U 604.11 negieren.

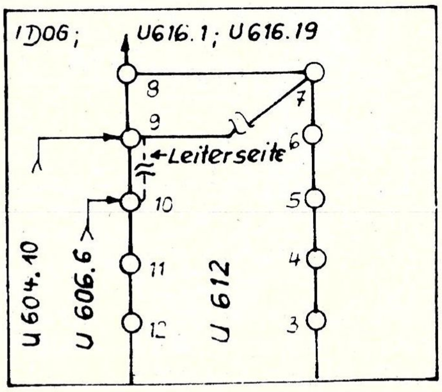


Abb. 12 Bestükungsseite NAND-Gatter U 612-3 freimachen und altes DOG mit /DIOG zu neuem /DOG verknüpfen.

Pin 13 ––> U 607 Pin 2 –– > U 612 Pin 5 ––> U 606 Pin 4 ––> U 603 Pin 5 ––> U 602 Pin 9 wird vor und hinter U 606 Pin 4 aufgetrennt, und die beiden Teilleiterzüge werden mit einer Brücke wieder zusammengesetzt und liefern das Signal CE (/AO). Pin 4 der IS U 606 erhält das Ausgangssignal CA.

Abbildung 7 hilft Ihnen bei dieser Änderung, die Brücke ist leichter auf der Leiterseite anzubringen.

c) Erzeugung der Signale DE und DA (Punkt D)

Leitung /GS vom Ausgang der IS U 508 Pin 12 direkt neben der Durchkontaktierung am Zwischenkonnektor auftrennen. Pin 12 von IS U 508 wird Eingangssignal DE und die Durchkontaktierung erhält das Ausgangssignal DA (/GS).

Abbildung 8 erleichtert Ihnen die Suche nach der Durchkontaktierung.

d) Erzeugung der Signale EE und EA (Punkt E)

Das Ausgangssignal CRUCLK der CPU wird unmittelbar hinter Pin 60 der CPU auf der Leiterseite getrennt. Pin 60 wird zum Signal EE (CPU-CRUCLK) und der abgetrennte Leiterzug erhält das Signal EA (CRUCLK).

Abbildung 9 erläutert diese Änderung.

e) Erzeugung der Verriegelung zwischen U 614 und 616 (Punkt F)

Diese Verriegelung braucht nur erzeugt zu werden, wenn man das Eingangssignal /EXIO zur Beschleunigung von 8-Bit-Zugriffen außerhalb der Konsole benutzen will!

Wenn die Verriegelung nicht vorgenommen wird, darf der Eingang /EXIO auf keinen Fall aktiviert werden!

Diese Änderungen können in jedem Fall durchgeführt werden, sie stören nicht, auch wenn vorläufig nicht der Eingang /EXIO benutzt werden soll.

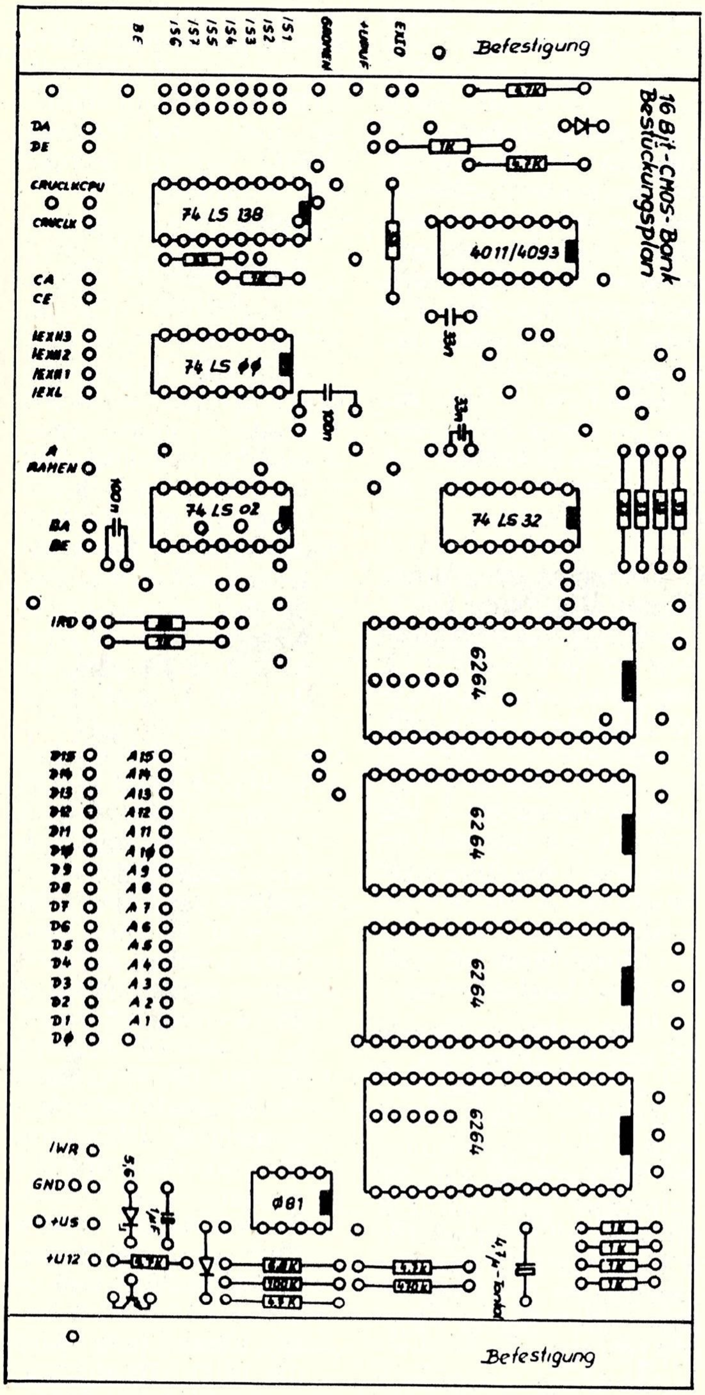
Änderungen an der Konsole

Folgende Änderungen sind auf der Platine der Konsole vorzunehmen:

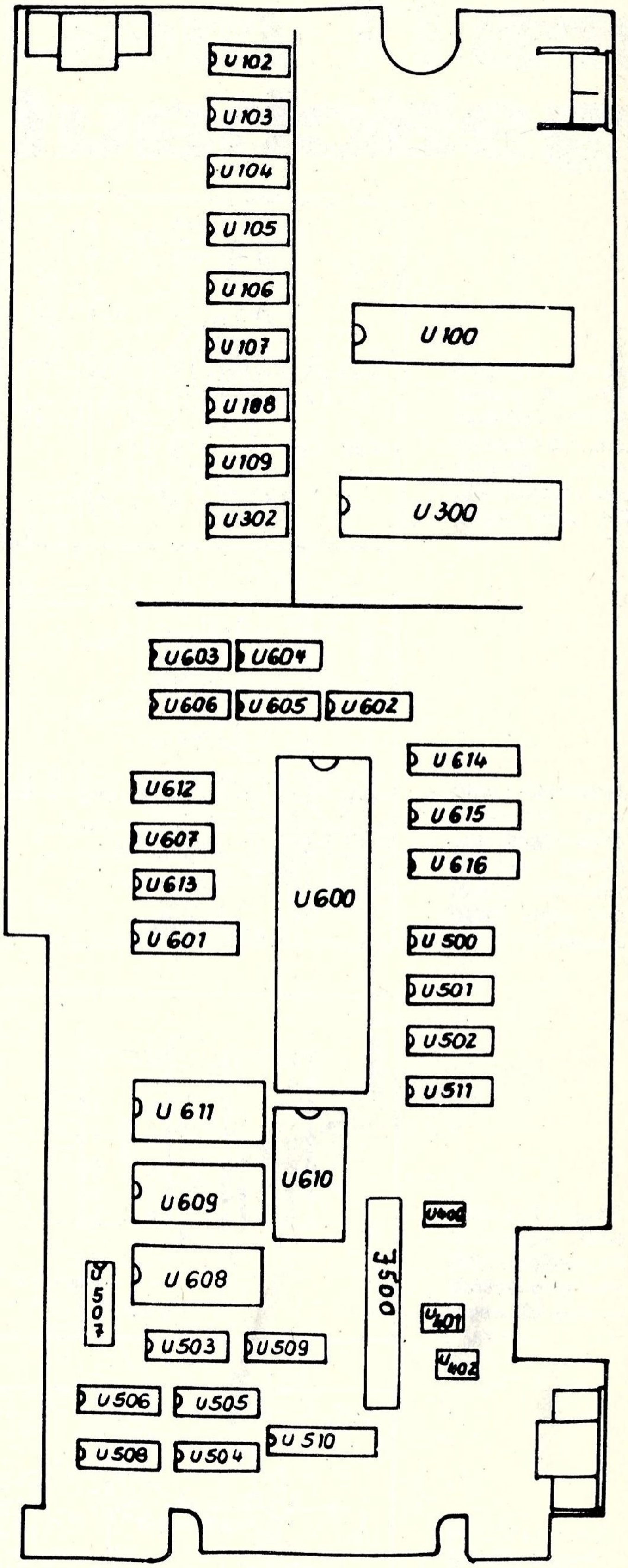
– Das Signal /DOG wird von Pin 1 und 19 der IS U 616 (Punkt F in Abbildung 1) abgetrennt und statt dessen auf den Eingang 11 der Negator-IS U 604 gegeben. Der Ausgang 10 der IS U 604 führt dann zum Eingang Pin 9 des NAND-Gatters IS U 612. Diese beiden Eingänge (Pin 9 und 10) liegen zur Zeit über einer Brücke unterhalb der IS auf Masse. Die IS muß entfernt werden, der Leiterzug aufgetrennt und die IS wieder eingesetzt werden. Eingang 10 des NAND-Gatters U 612 erhält das Selektionssignal /DIOG von IS U 606 Pin 6 und der Ausgang des NAND-Gatters U 612 Pin 8 führt nun das Signal /DOG zu Pin 1 und 19 der IS 616. Durch diese Verriegelung wird die IS U 616, die normalerweise bei jedem Schreibzyklus mit aktivem Adreßbit AB0 aktiv ist, immer dann abgeschaltet, wenn bei Umgehung der Ablaufsteuerung ein schneller Zugriff über die IS U 614 durch Aktivierung des Eingangs /EXIO erfolgt.

f) Mechanische Änderungen

Da die Zusatzleiterkarte direkt unter die Konsolenleiterkarte montiert



Bestückungsplan der EVOLUTION - Platine



wird, ist in die Schirmung eine entsprechend große Aussparung einzuarbeiten, was aber ohne Probleme möglich ist und auch die Störstrahlungsverhältnisse nicht erheblich verschlechterte. Im Muster wird die Konsole ohnehin wieder auf einer schirmenden Montagefläche zusammen mit Netzteil, Audioverstärker und PAL-Modulator sowie RGB- bzw. Video-Ansteuerung untergebracht, sodaß ein Gesamtgerät ohne den Aufwand ständigen Zusammensteckens der Komponenten entsteht. Die Stromversorgung dieses Gerätes erfolgt durch eine rückwärtige Kaltgeräte-Schutzkontakt-Steckdose durch Versorgung aus der Erweiterungsbox, sodaß nur noch ein Schalter zu betätigen ist. Der Konsolen-„Ein/Aus“-Schalter wird nicht mehr benutzt.

Heinrich Cronjaeger■

Adresse	0141 Name : SNSMAT
Verändert	AF
Parameter	Tastaturreihe in A
Ergebnis	Status in A
Funktion	Gibt den Status einer bestimmten Tastenreihe zurueck
Bemerkung	Das einer Taste entsprechende Bit ist auf 0 gesetzt, wenn die Taste gedrueckt wird.

Adresse	0144 Name : PHYDIO
Verändert	
Parameter	
Ergebnis	
Funktion	Sektorweiser Zugriff auf Diskette
Bemerkung	Nicht dokumentiert

Adresse	0147 Name : FORMAT
Verändert	
Parameter	
Ergebnis	
Funktion	Formatieren einer Diskette
Bemerkung	Springt lediglich Hook FORMAT an.

Adresse	014A Name : ISFLIO
Verändert	AF
Parameter	
Ergebnis	Zero=0: Nein, Zero gesetzt: Ja
Funktion	Prueft, ob augenblicklich auf Peripherie zugegriffen wird
Bemerkung	

Adresse	014D Name : OUTDLP
Verändert	Flags
Parameter	Zeichen in A
Ergebnis	
Funktion	Ausgabe auf Drucker
Bemerkung	TABs und Grafikzeichen werden in Spaces verwandelt, bei Abbruch Aufruf der Device I/O-Error Routine

Adresse	0150 Name : GETVCP
Verändert	
Parameter	A: Nr. der Queue
Ergebnis	HL = Pointer
Funktion	Verwendung im PLAY-Statement
Bemerkung	

Adresse	0153 Name : GETVC2
Verändert	AF
Parameter	L: gewuenschte Plazierung
Ergebnis	HL = neuer Pointer auf Queue
Funktion	Verwendung im PLAY-Statement
Bemerkung	

Adresse	0156 Name : KILBUF
Verändert	HL
Parameter	
Ergebnis	
Funktion	Leert den Tastatur-Puffer
Bemerkung	Nuetzlich, um "Vorlaufen" der Tastatur zu unterbinden. Wird von INLIN etc. automatisch aktiviert.

Adresse	0159 Name : CALBAS
Verändert	Alle
Parameter	Adresse in IX
Ergebnis	
Funktion	fuehrt Far_CALL in den BASIC-Interpreter aus
Bemerkung	Kann auch aufgerufen werden, wenn man nicht genau weiss, an welcher Slot-Adresse der Interpreter zu finden ist.

Adresse	015C Name : SUBROM
Verändert	AF',IY
Parameter	Adresse in IX, Kopie von IX auf Stack
Ergebnis	??
Funktion	Fuehrt Far_Call in Subrom aus
Bemerkung	Nur MSX 2.Eine von zwei Moeglichkeiten, das Subrom aufzurufen

Adresse	015F Name : EXTROM
Verändert	AF', IY
Parameter	Adresse in IX
Ergebnis	??
Funktion	Fuehrt Far_Call in das Subrom aus
Bemerkung	Nur MSX 2.-Zweite Moeglichkeit

Adresse	0162 Name : CHKSLZ
Verändert	Alle
Parameter	
Ergebnis	
Funktion	Durchsucht Slots nach Subrom
Bemerkung	Nur MSX 2

Adresse	0165 Name : CHKNEW
Verändert	AF
Parameter	
Ergebnis	Carry zurueckgesetzt bei Screen 5..8
Funktion	Prueft Bildschirm-Modus
Bemerkung	Nur MSX 2

Adresse	0168 Name : EOL
Verändert	Alle
Parameter	Spalte in H, Zeile in L
Ergebnis	
Funktion	Loescht bis zum Zeilenende
Bemerkung	Nur MSX 2

Adresse	016B Name : BIGFIL
Verändert	Wie FILVRM
Parameter	Wie FILVRM
Ergebnis	Wie FILVRM
Funktion	Wie FILRVRM, allerdings Adressen bis 64 KByte zulaessig
Bemerkung	Nur MSX 2. Bei Screen 0..3 exakt die gleiche Funktion wie FILVRM

Adresse	016E Name : NSETRD
Verändert	AF
Parameter	Adresse in HL: alle Bits gueltig
Ergebnis	
Funktion	Bereitet VDP zum Lesen vor
Bemerkung	Nur MSX 2

Adresse	0171 Name : NSTWRT
Verändert	
Parameter	Wie NSETRD
Ergebnis	
Funktion	Bereitet VDP zum Schreiben vor
Bemerkung	Nur MSX 2

Adresse	0174 Name : NRDVRM
Verändert	
Parameter	Adresse in HL
Ergebnis	Wert in A
Funktion	"Vpeek" an Adresse HL. Alle Bits gueltig
Bemerkung	Nur MSX 2

The Castle of Dr. Hype

Insgesamt gibt es sechs Räume, die es zu durchwandern gilt, um den Ausgang zu finden. Man steuert seine Spielfigur mittels Joystick 1. Gefahr droht von den Löchern im Boden, von elektrostatischen Generatoren u.v.a.

DAS SPIEL

Das Ziel besteht nun darin, in jedem Bild den Ausgang, falls verschlossen, zu öffnen, um sich in den nächsten Raum zu begeben.
Um überhaupt an eine Tür zu gelangen, müssen Leitern und Stricke herhalten, auch die sogenannten Beamkabinen (siehen aus wie Waschmaschinen). Besteigt man eine solche, verschwindet man und taucht an einer anderen bestimmten Stelle wieder auf. Jedoch gibt es da noch die elektrostatischen Generatoren, die ihre tödlichen Energieblitze auf Sie schleudern. Mittels kleinen Hebeln (durch Drauffahren zu betätigen) können sie abgeschaltet werden. Doch Vorsicht, manche Hebel haben auch völlig andere Wirkungen.
Sie sterben, wenn Sie in die Energieblitze hineinfahren oder wenn direkt unter Ihnen nichts mehr ist, zum Beispiel bewegen Sie sich auf einer Leiter nach links. Dort aber ist gähnende Leere. Jetzt dürfen Sie auf keinen Fall versuchen, zurückzufahren, Ihre Spielfigur würde nämlich sofort abstürzen. Daher ist in einem solchen Fall der Feuerknopf zu drücken. So werden Sie an einen anderen Platz teleportiert. Sollten Sie allerdings sterben (Sie haben nur ein Leben), müßten Sie wieder am Anfang beginnen, und Ihre bisher erreichten Leistungen wären dahin.
Also: Ein Loch unter der Spielfigur besagt, drücke die Taste nur.
In diesem Sinne: Glückauf, und staunen Sie, was der TI noch alles drauf hat.

```
100 ! **********************
110 ! *                    *
120 ! * The Castle of      *
130 ! *      Dr. Heipe     *
150 ! *    Copyright by    *
160 ! *   Peter  Naumann   *
170 ! *                    *
190 ! * Benoetigte Geraete *
200 ! *  TI99/4A  Konsole  *
210 ! *     Ext. Basic     *
220 ! *     Joystick 1     *
230 ! *                    *
260 ! *  Speicherbelegung  *
270 ! *     10719 Bytes    *
280 ! *                    *
290 ! **********************
300 GOSUB 1660
310 CALL CHAR(80,"0000000000
071F2F183C3C3CFFE70000000000
0000E0F8F427203E5D5D5D5D3E")
320 CALL CHAR(84,"C324DB9966
241800E4047CBABABABA7C142A3E
5D5D5D5D3E28547CBABABABA7C")
330 QW=0
340 CALL CHAR(33,"FF8181FF99
9981FFFF819999FF8181FF427E42
42427E42427F5555555555557F")
350 CALL CHAR(37,"423C423C42
3C423C10101010101010100000007
FDA7")
360 CALL CHAR(40,"00FCFCFCFC
FCFCFC",42,"7FBFDFEFF7FBFDFE
BDBDBDDBDB810000FEFDFBF7EFDF
BF7F003F3F3F3F3F3F3F")
370 CALL CHAR(47,"000000030F
1F7FFFFFFFFFFFFFFFF00FF0F1F3F
7FFFFF80FFFFFEFCF8F0E020E0")
380 CALL CHAR(50,"FFFFFDF9F2
E428F01C2C5CFCBCBCBCBCBFBFBF
BFBFFF80FF")
390 CALL CHAR(56,"0F153F55FF
FFFFF3FE56FA52E2E2E2E2E1E1F3
FFFFFFFFFFE2E2E2E2E4E8F0E")
400 CALL CHAR(64,"187E6EDFDF
6E7E18222414181824 2242")
410 CALL COLOR(1,11,1,2,5,1,
3,3,1,4,15,1,5,9,1)
420 CALL COLOR(1,11,1,2,5,1,
3,3,1,4,15,1,5,9,1)
430 CALL COLOR(9,10,1)
440 QW=0
450 CALL CLEAR :: CALL SCREE
N(2)
460 CALL CHAR(96,"1038107CBA
A92828")
470 DISPLAY AT(1,1):" *+,
                  *+,   ( -
                  ( -  ($-  #
             #    (/- 100002#
12     #      12#100002"
480 DISPLAY AT(5,1):"       #
    12#12      #       89   #
       #       #       :;   #
       #       #     1000000
02  12#12    10212 12"
490 DISPLAY AT(9,1):"
       #

     !  #  !
 10002#10002
   @   #  @"
500 DISPLAY AT(13,1):"
  # A   #   A #
12#100000002#12
  #  @   89   #  89        3
! #   A   :;   # !:;"
510 DISPLAY AT(17,1):"     40
0000000002 100002
                  @ 89
#     # !    !#   #A :;     102
#1002#102 12#   #10002"
520 DISPLAY AT(21,1):"     @
#     # @     #@ #       3  A
#     # A     #A #   !   40000
002 1000000000000002"
530 V=3 :: B=5 :: RAUM=1
540 CALL CHAR(65,"1028244244
844428"):: CALL GCHAR(V,B,Z)
:: CALL HCHAR(V,B,96)
550 CALL JOYST(1,X,Y):: CALL
 CHAR(65,"081424422212214")
:: CALL KEY(1,K,S):: IF K=18
 THEN 1490
560 IF X+Y=0 THEN 550
570 CALL HCHAR(V,B,Z)
580 CALL GCHAR(V,B,Z1):: IF
Z1=47 THEN 1180 :: CALL GCHA
R(V+1,B,Z2):: IF Z2=32 THEN
1170 :: IF Z1=51 THEN B=B+1
590 IF Z1=58 THEN 630 ELSE I
F Z1=33 THEN 860 ELSE IF Z1=
65 THEN 1170 ELSE IF Z1=39 T
HEN 1470
600 B=B-((X=-4)*(Y=0))+((X=4
)*(Y=0))-(B=2)+(B=31):: IF Z
1=35 OR Z2=35 OR Z1=38 OR Z2
=38 THEN V=V-((X=0)*(Y=4))+(
(X=0)*(Y=-4))-(V=2)+(V=23)
610 CALL SOUND(100,44733,0)
620 GOTO 540
630 FOR N=0 TO 20 STEP 3 ::
CALL SOUND(-100,-5,N):: CALL
 SOUND(-100,-5,N+10):: CALL
SOUND(-100,-5,N+5):: NEXT N
640 IF RAUM<>2 THEN 710
650 IF V=22 AND B=12 THEN V=
17 :: B=14 :: GOTO 540
660 IF V=22 AND B=17 THEN V=
```

```
14 :: B=19 :: GOTO 540
670 IF V=8 AND B=10 THEN V=3
 :: B=9 :: GOTO 540
680 IF V=12 AND B=22 THEN V=
18 :: B=22 :: GOTO 540
690 IF V=3 AND B=5 THEN V=3
:: B=18 :: GOTO 540
700 IF V=3 AND B=15 THEN V=3
 :: B=9 :: GOTO 540
710 IF RAUM<>1 THEN 760
720 IF V=16 AND B=24 THEN B=
13 :: GOTO 540
730 IF V=16 AND B=17 THEN B=
11 :: V=19 :: GOTO 540
740 IF V=19 AND B=27 THEN V=
7 :: B=25 :: GOTO 540
750 IF V=7 AND B=4 THEN V=4
:: B=15 :: GOTO 540
760 IF RAUM<>3 THEN 810
770 IF V=22 AND B=27 THEN V=
13 :: B=21 :: GOTO 540
780 IF V=13 AND B=27 THEN V=
17 :: B=21 :: GOTO 540
790 IF V=17 AND B=27 THEN V=
9 :: B=21 :: GOTO 540
800 IF V=9 AND B=27 THEN V=5
 :: B=11 :: GOTO 540
810 IF RAUM<>4 THEN 540
820 IF V=8 AND B=29 THEN V=5
 :: B=25 :: GOTO 540
830 IF V=5 AND B=23 THEN V=2
2 :: B=15 :: GOTO 540
840 IF V=5 AND B=18 THEN V=8
 :: B=16 :: GOTO 540
850 IF V=22 AND B=16 THEN V=
5 :: B=20 :: GOTO 540
860 CALL SOUND(800,-8,0):: C
ALL HCHAR(V,B,34)
870 IF RAUM<>2 THEN 960
880 IF V=12 AND B=7 THEN CAL
L HCHAR(22,11,32)
890 IF V=17 AND B=17 THEN CA
LL HCHAR(22,23,32)
900 IF V=14 AND B=14 THEN CA
LL HCHAR(8,12,32)
910 IF V=8 AND B=5 THEN CALL
 HCHAR(22,26,32)
920 IF V=3 AND B=17 THEN CAL
L HCHAR(8,6,32)
930 IF V=3 AND B=30 THEN CAL
L HCHAR(18,23,32)
940 IF V=6 AND B=30 THEN CAL
L HCHAR(22,24,32)
950 IF V=14 AND B=27 THEN CA
LL HCHAR(22,25,32)
960 IF RAUM<>1 THEN 1040
970 IF V=10 AND B=19 THEN CA
LL HCHAR(13,13,32)
980 IF V=10 AND B=13 THEN CA
LL HCHAR(13,19,32)
990 IF V=16 AND B=9 THEN CAL
L HCHAR(22,7,32)
1000 IF V=16 AND B=23 THEN C
ALL HCHAR(16,14,32)
1010 IF V=19 AND B=16 THEN C
ALL HCHAR(22,16,32)
1020 IF V=19 AND B=20 THEN C
ALL HCHAR(22,22,32)
1030 IF V=22 AND B=27 THEN C
ALL HCHAR(19,25,32)
1040 IF RAUM<>3 THEN 1060
1050 IF V=13 AND B=24 THEN C
ALL HCHAR(5,13,32):: GOTO 54
0
1060 IF V=4 AND B=7 THEN DIS
PLAY AT(17,21)SIZE(3):"000"
:: GOTO 540
1070 IF V=4 AND B=10 THEN CA
LL HCHAR(16,18,32):: GOTO 54
0
1080 IF V=5 AND B=21 THEN DI
SPLAY AT(17,18)SIZE(3):"000"
 :: GOTO 540
1090 IF V=9 AND B=6 THEN CAL
L HCHAR(22,10,32):: GOTO 540
1100 IF V=13 AND B=6 THEN CA
LL HCHAR(18,10,33):: GOTO 54
0
1110 IF V=18 AND B=10 THEN C
ALL HCHAR(13,7,33):: GOTO 54
0
1120 IF V=13 AND B=7 THEN CA
LL HCHAR(16,19,32):: GOTO 54
0
1130 IF V=6 AND B=11 THEN CA
LL HCHAR(4,8,32)
1140 !
1150 CALL JOYST(1,X,Y):: IF
X+Y=0 THEN 1150
1160 B=B-(K=68)+(K=83):: GOT
O 540
1170 CALL CHAR(96,"18187E7E1
8181818"):: CALL HCHAR(V,B,9
6):: GOTO 420
1180 IF RAUM=1 AND V=3 AND B
=28 THEN 1230 ELSE IF RAUM=2
 AND V=22 AND B=29 THEN
1370 ELSE IF RAUM=3 AND V=19
 AND B=6 THEN 1230
1190 IF RAUM=2 AND V=3 AND B
=23 THEN 1370
1200 IF RAUM=3 AND V=5 AND B
=24 THEN CALL HCHAR(22,13,32
):: CALL HCHAR(V,B,36):: CAL
L HCHAR(22,17,47):: V=5 :: B
=11 :: GOTO 540
```

```
1210 IF RAUM=3 AND V=22 AND
B=17 THEN 1540
1220 IF RAUM=4 THEN 1630
1230 GOTO 1270
1240 FOR N=0 TO 12 :: CALL S
OUND(-150,329,N):: NEXT N
1250 FOR N=0 TO 9 :: CALL SO
UND(-150,277,N,329,N+11):: N
EXT N
1260 FOR N=0 TO 9 :: CALL SO
UND(-150,220,N,277,N+11,329,
N+21):: NEXT N :: RETURN
1270 GOSUB 1240 :: CALL CLEA
R
1280 IF QW=1 THEN CALL COLOR
(1,15,1,2,7,1,3,11,1,4,4,1,5
,10,1)ELSE CALL COLOR(1,7,1,
2,14,1,3,6,1,4,3,1,5,11,1)
1290 DISPLAY AT(1,1):"
             *+,        89
      89     ( -        :; #
      :;!  # ($-   # ! 1002#
12  1000002#1002 12#12"
1300 DISPLAY AT(5,1):"      #
           #       #       #
           #    #  # !   @ #
 89@       #  12#12#12   !A #
 :;A #     #    #  #"
1310 DISPLAY AT(9,1):" 10000
00002#12 121212102 #
     #             #
     # #     89 #  #    3 !
#   102#12   :; #  #"
1320 DISPLAY AT(13,1):"  400
2#12    #    1002#  #
 #    ! #       # !#
 #  100000002   100002
 #"
1330 DISPLAY AT(17,1):" 3
 #     & !     @        4000
2#12 12&12   # A !
 #     &   12#10002     *+,
 #     &     # % %  *+,"
1340 DISPLAY AT(21,1):" ( -
 # @89 & 89  # @@@@ ( - ($-
 # A:; & :;  # AAAA (/-10000
00000000002 10000000002"
1350 RAUM=2 :: V=22 :: B=5 :
: IF QW=1 THEN CALL HCHAR(3,
23,47):: CALL HCHAR(22,29,36
)
1360 GOTO 540
1370 GOSUB 1240 :: CALL CLEA
R
1380 IF QW=1 THEN CALL COLOR
(1,5,1,2,11,1,3,7,1,4,13,1,5
,4,1)ELSE CALL COLOR(1,11,1,
2,7,1,3,3,1,4,6,1,5,16,1)
1390 DISPLAY AT(3,1):"
              *+,
    @         ( -"
1400 DISPLAY AT(5,1):"
#   A        ($-       10002
#10000000000000000002
#   %
#   %             89"
1410 DISPLAY AT(9,1):"
#   %  3          :;
#   %  40002100000002  10002
#12 %
#   %             89"
1420 DISPLAY AT(13,1):"
 #   %  3       !  :;      '
 #   %  40000000000002  1210
2#12 %
 #   %             89"
1430 DISPLAY AT(17,1):"  *+,
 #   %  3          :;    ( -
 #   %  40000021210002   ($-
 #   %                  1000
2#12 %  *+,"
1440 DISPLAY AT(21,1):"
 #   @  ( -        89
 #   A  ($-        :;   1000
0000000000000000000000002"
1450 RAUM=3 :: IF QW=1 THEN
CALL HCHAR(5,24,47)
1460 QW=1 :: V,B=22 :: GOTO
540
1470 IF RAUM=3 THEN CALL HCH
AR(19,6,47):: V=14 :: B=7 ::
 GOTO 540
1480 IF RAUM=4 THEN CALL HCH
AR(5,27,47):: CALL HCHAR(8,2
8,32):: GOTO 540
1490 CALL HCHAR(V,B,Z)
1500 IF RAUM=1 THEN V=3 :: B
=5 :: GOTO 540
1510 IF RAUM=2 THEN V=22 ::
B=5 :: GOTO 540
1520 IF RAUM=3 THEN V=22 ::
B=22 :: GOTO 540
1530 IF RAUM=4 THEN V=22 ::
B=8 :: GOTO 540
1540 GOSUB 1240 :: CALL CLEA
R
1550 CALL COLOR(1,7,1,2,5,1,
3,3,1,4,15,1,5,6,1,6,9,7,7,1
1,1,10,10,1,11,10,1)
1560 DISPLAY AT(3,1):"      @
                 PQR       !A
 !  #    89   89 STU"
1570 DISPLAY AT(5,1):"   100
0002#12  :; ! :; V$W
```

```
  ! #   121212100002
 102#               89
    #   #         &':;"
1580 DISPLAY AT(9,1):"    !
 &  # 12#12   12  &102  1000
2&  #   #    12   &
 &  #   #   12    &
 &  #  ##     12  &"
1590 DISPLAY AT(13,1):"    !
& &  #  ##12  12  102#   102
&102 #  #    12      #
&    #  # @@   12    #
&    #  # AA       # #"
1600 DISPLAY AT(17,1):"    &
&    # 100212 12 12# #    &
&    #             # #  12&1
2102 #        12   # #    &
  %  #       12    # #"
1610 DISPLAY AT(21,1):"    &
  @  #  89     12  # #    &
  A  #  :|    12   # #
1212100000000002 12   10002"
1620 RAUM=4 :: V=13 :: B=9 :
: GOTO 540
1630 !
1640 FOR Q=110 TO 2200 STEP
50 :: CALL SOUND(-99,Q,0,Q+9
,0,Q+18,0):: NEXT Q
1650 CALL CLEAR :: END
1660 CALL CHAR(40,"010303010
1397FCFC060603030180F0780C0C
080809CFEF30306060C0C18F0E",
44,"0000000C1E3F7FFF00000000
000080C")
1670 CALL CHAR(48,"3F3F3F332
1212121003 3FFFFBDFFFFFFDBFFF
FDBFF7E7E7E7E7EFFFFBDFFFFFF"
)
1680 CALL CHAR(52,"000000000
00000CCFCFCFCCC84848484704C4
34C7040404000303C3000000000"
)
1690 CALL CLEAR
1700 CALL SCREEN(4)
1710 CALL COLOR(2,7,1,3,2,1,
5,3,1,6,3,1,7,3,1,8,3,1)
1720 DISPLAY AT(1,5):"(*  (*
":TAB(5);")+(*)+":TAB(7);")+
"
1730 DISPLAY AT(5,3):"THE CA
STLE OF"
1740 DISPLAY AT(13,19):",-24
 ,-24":TAB(19);"0135 0135"
1750 CALL MAGNIFY(2)
1760 CALL SPRITE(#1,68,15,65
,33,#2,79,15,57,49,#3,67,15,
49,65,#4,84,15,49,89,#5,79,1
5,57,105)
1770 CALL SPRITE(#6,82,15,65
,121,#7,72,16,97,45,#8,69,16
,89,61,#9,73,16,81,77,#10,80
,16,89,93)
1780 CALL SPRITE(#11,69,16,9
7,109)
1790 FOR Q=1 TO 11 :: CALL M
OTION(#Q,5,0)
1800 FOR N=200 TO 110 STEP -
10 :: CALL SOUND(-100,N,5)::
 NEXT N
1810 CALL MOTION(#Q,0,0):: N
EXT Q
1820 FOR Q=1 TO 11 :: CALL M
OTION(#Q,-5,0)
1830 FOR N=110 TO 200 STEP 1
0 :: CALL SOUND(-100,N,5)::
NEXT N
1840 CALL MOTION(#Q,0,0):: N
EXT Q
1850 CALL SPRITE(#12,54,9,32
,161,#13,55,11,32,161,#14,54
,9,16,185,#15,55,11,16,185,#
16,54,9,32,209)
1860 CALL SPRITE(#17,55,11,3
2,209)
1870 FOR Q=1 TO 10
1880 CALL CHAR(54,"7E4448506
04040400038302")
1890 GOSUB 1990
1900 CALL CHAR(54,"704C434C7
0404040000303C3")
1910 GOSUB 1990
1920 CALL CHAR(54,"605048447
E40404000203038")
1930 GOSUB 1990
1940 CALL CHAR(54,"704C434C7
0404040000303C3")
1950 GOSUB 1990
1960 NEXT Q
1970 DISPLAY AT(19,1)SIZE(22
):"PRESS ANY KEY TO BEGIN"
1980 CALL KEY(0,K,S):: CALL
KEY(1,K1,S1):: IF S=0 AND S1
=0 THEN 1980 ELSE 2020
1990 A=INT(RND*44620)+110
2000 CALL SOUND(-1000,-8,15,
A,30,A,30,A,30)
2010 RETURN
2020 E=110
2030 E=E*2^(1/12)
2040 IF E>44733 THEN CALL CL
EAR :: CALL DELSPRITE(ALL)::
 RETURN
2050 CALL SOUND(-250,E,30,E,
30,E,30,-8,0)
2060 GOTO 2030
```

Dem TI-unters Hemd geschaut

Durch die Erhöhung der Arbeitsgeschwindigkeit des 16-Bit-breit angelegten RAM-Bereiches kommt es bei Maschinenprogrammen, die auf GROM-Routinen zurückgreifen, zu Zeitproblemen mit den GROMs. Diese ROMs sind sehr langsam und benötigen eine Erholzeit zwischen zwei Zugriffen für eine ordnungsgemäße Ausführung aller Zugriffe (offensichtlich benötigen die GROMs diese Zeit zur Synchronisation mit GROMCLK). Daher sind in den Routinen mit aufeinanderfolgenden GROM-Zugriffen auch NOPs oder andere Befehle in die Zugriffe eingeschoben, damit diese Zeitprobleme beseitigt werden. Da die Maschinen-Programme jetzt doppelt so schnell ablaufen, können diese Zeitbedingungen verletzt werden. Das ist zum Beispiel der Fall, wenn auf die höheren mathematischen GROM-Routinen zurückgegriffen wird. Hierfür gibt es mehrere Auswege:

- man schreibt die Programme selber, dann kann man diese Zeitprobleme berücksichtigen;
- man greift nicht auf GROM-Routinen zurück;
- man erzeugt künstlich nach GROM-Zugriffen die notwendige Erholzeit;
- man benutzt die universelle GROM-GRAM-Steckeinheit, die im Abschnitt 3, REVOLUTION, beschrieben wird, dann treten diese Zeitprobleme gar nicht erst auf. Im Gegenteil, dann werden die GROM-Routinen noch beschleunigt.

Es hat schon Berichte gegeben, daß solche Zeitprobleme auch bei Zugriffen auf den Videoprozessor auftreten können, solche Erfahrungen kann der Autor aber nicht teilen. Selbst speziell auf hohe Videoprozessor-Zugriffsrate zugeschnittene Testprogramme konnten den Videoprozessor nicht „überfahren", so daß derartige Bedenken nicht gerechtfertigt erscheinen. Aber an dieser Stelle soll gezeigt werden, daß es selbst für diesen Fall einen Ausweg ohne spürbare Zeitbilanzverluste gibt. *Abbildung 15* zeigt eine kleine Hilfsschaltung, mit der die „kritischen" Buszyklen erkannt werden und der Prozessor durch Aktivierung des Signals /HOLD für eine einstellbare Zeit (ca. 4 bis 5 Mikrosekunden) gestoppt wird. Diese Aus-Zeit genügt den GROMs

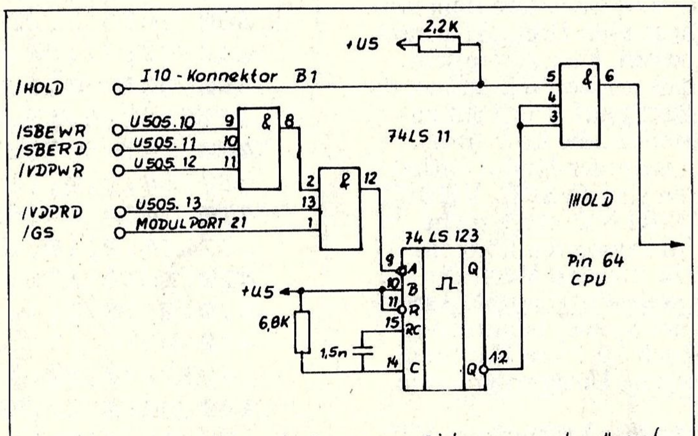


Abb. 15 Zusatzbeschaltung zur Sicherung schnell aufeinander folgender GROM- und VDP-Zugriffe

Abbildung 15:

zur Erholung. In der Abbildung wird auch die Erfassung der VDP-Zugriffe vorgenommen, nach Meinung des Autors besteht dafür aber keine Notwendigkeit. Für andere Erfahrungen und Hinweise zu dieser Problematik wäre er jedoch dankbar.

ERLÄUTERUNGEN ZU DER „SICHERHEITS"-SCHALTUNG:

Mit dem monostabilen Multivibrator (74LS123) wird bei Erkennung eines „kritischen" Buszyklus ein low-aktiver Impuls von ca. 4,6 Mikrosekunden (Einstellung zwischen 4 und 5 Mikrosekunden ist ausreichend) erzeugt, der das /HOLD-Signal für die CPU (Pin 64) erzeugt. Nach Abschluß des gerade laufenden Buszyklus wird die CPU ihren Betrieb einstellen und auf /HOLD = high warten, dann wird sie ihren Betrieb fortsetzen. Dadurch wird künstlich die Zeitverzögerung erzeugt, die sonst der Befehlsholezyklus auf der Acht-Bit-organisierten Speichererweiterung benötigt hätte.

Zur Schaltung selbst: Der 74LS123 wird direkt über IS U 504 (74LS138) montiert, alle „Beinchen" außer der Betriebsspannung werden gekürzt, Pin 8 und 16 werden an den entsprechenden Pins der U 504 angelötet. Der Widerstand (6,8 KByte) und der Kondensator (1,5 nF) werden direkt am Schaltkreis montiert und ergeben eine Zeitkonstante für den /HOLD-Impuls von ca. 4 bis 5 Mikrosekunden. Es wird nur der Eingang A genutzt. Eingang B und /RESET werden auf high festgelegt. Das zweite System des LS123 wird nicht benutzt und die Eingänge auf high gelegt.

Mit dem 74LS11 werden an den Eingängen 1, 10, 11, 12 und 13 die eventuell kritischen Buszyklen /GS, /SBEWR, /SBERD, /VDPWR und /VDPRD erfaßt und an den Eingang /A des 74LS 123 weitergegeben. /GS wird direkt am Zwischenkonnektor zum Modulport abgenommen, damit bei Verwendung der im Abschnitt "REVOLUTION" beschriebenen GROM-GRAM-Karte die /HOLD-Erzeugung unterdrückt wird (dann sind die internen GROMs abgeschaltet). Die restlichen Signale werden direkt am Dekoder U 505 abgenommen. Dazu wird auch der 74LS11 direkt über dem U 505 montiert, und zwar mit Pin 7 des 74LS11 über Pin 8 des U 505. Dadurch lassen sich die Verbindungen zu /SBEWR, /SBERD und VDPWR direkt und zu /VDPRD durch Verbinden zum Nachbarpin (74LS11.13 zu U 505.13) herstellen. Der Pull-Up-Widerstand von 2,2 K-Byte an Pin 5 wird vorher montiert. An diesem Pin kann über den I/O-Steckverbinder B1 ein /HOLD erzeugt werden, da die Konsole sonst keine /HOLD-Generierung erlaubt. Pin 8 wird zu Pin 2 gebrückt und Pin 12 führt zu Pin 9 des 74LS123 (Eingang A). Der Ausgang /Q des 74LS123 wird mit Pin 3,4 des 74LS11 verbunden und die Verbindung von I/O-Konnektor B1 zum Pin 5 des 74LS11 hergestellt. Und fertig ist die „Hilfsschaltung".

Liebe TI-Fans! Mit dieser EVOLUTION der TI-Konsole haben wir die Leistungsfähigkeit des Rechners schon erheblich gesteigert. Richtig rund gehen wird es aber erst in dem folgenden Beitrag REVOLUTION, wo die Leistungsfähigkeit des ehemals von TI verfolgten Konzeptes nicht nur voll erreicht wird, sondern in der Verarbeitungsgeschwindigkeit noch überboten wird und neue Anwendungsmöglichkeiten eröffnet werden. □

Disc to Tape

Nachdem im TI-Teil der HCA 7/88 ein Programm abgedruckt war, mit dem man Files im Memory Image-Format (sogenannte PROGRAM-Files) von Kassette unter Extended-BASIC einlesen kann, folgt logischerweise nun das Programm, mit dem man diese Files von Diskette auf Kassette überspielen kann. Es ist also auch für diejenigen TI-Freaks, die nur über Ex-BASIC, Kassettenrecorder und eine 32-KByte-Erweiterung verfügen, möglich, auf eine Vielfalt von Programmen zurückzugreifen, die bisher zumindest Diskettenlaufwerk und das Editor/Assembler-Modul erforderten. Derartige Programmkombinationen werden schon seit einiger Zeit von verschiedenen „User-Clubs" und Privatleuten für relativ viel Geld angeboten – Leser der HCA können sich diese Ausgabe sparen.

Zum Programm selber: Nach Eingabe der POKE-Liste das Programm unbedingt abspeichern! Nach der Eingabe von RUN dauert es etwa 20 Sekunden, bis die Datas geladen sind. Nun wird das Programm mit CALL LINK("DSKCS1", Dateiname) aufgerufen. Ist die angesprochene Datei (ein PROGRAM-File) vorhanden, wird es geladen. Anschließend erscheinen die bekannten Aufforderungen zum Starten des Kassettenrecorders.

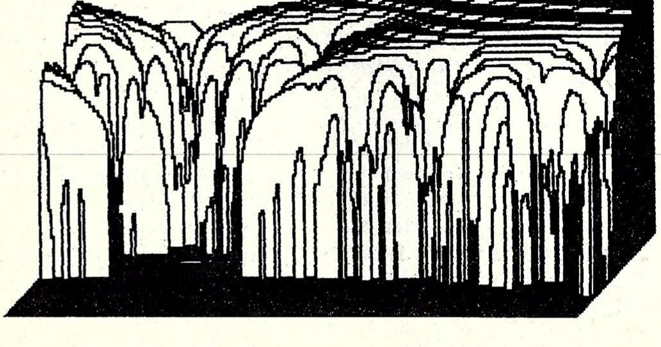
Pro Aufruf des Programms wird jeweils nur ein File auf CS1 abgespeichert. Besteht ein Programm aus mehreren PROGRAM-Files, müssen diese nacheinander durch einen jeweils erneuten Aufruf von DSKCS1 nachgeladen werden.

Der größte Teil der von mir zu Testzwecken ausprobierten Programme ließ sich ohne Probleme mit Hilfe der beiden Programme XLOADCS (HCA 7/88) und dem vorliegenden DSKCS1 verarbeiten. Klar dürfte sein, daß solche Programme, die direkt auf Routinen zugreifen, die im Editor/Assembler-Modul enthalten sind (VSBW, VMBW, DSRLNK, etc.), nicht funktionieren können. Die meisten Memory-Image-Programme (Assembler-Spiele) benutzen jedoch für diese Routinen eigene Unterprogramme.

Mathias Eichhorn
TI-Workshop Wiesbaden □

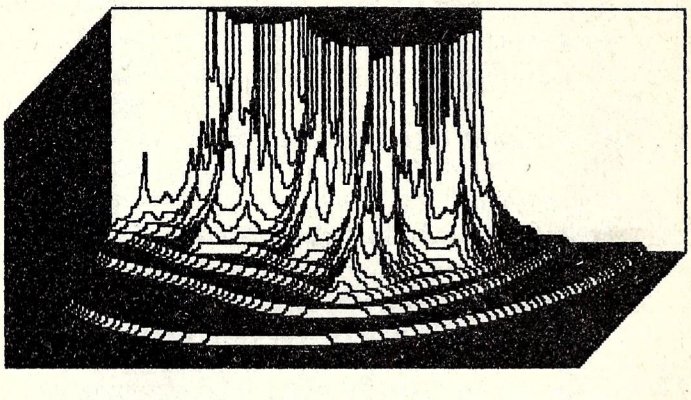
```
100 ! *********************
110 ! *                   *
120 ! * DSKCS1  POKELISTE *
130 ! *                   *
140 ! *     M. EICHHORN   *
150 ! *  GOETHESTRASSE 7  *
160 ! * 6242 KRONBERG/TS. *
170 ! *                   *
180 ! *********************
190 CALL INIT
200 ! Programm-Datas laden
210 FOR I=0 TO 80
220 READ A(1),A(2),A(3),A(4)
,A(5),A(6),A(7),A(8)
230 CALL LOAD(9460+I*8,A(1),
A(2),A(3),A(4),A(5),A(6),A(7
),A(8))
240 NEXT I
250 ! DEF-Table setzen
260 CALL LOAD(16376,68,83,75
,67,83,49,37,100)
270 ! Pointer auf DEF-Table
280 CALL LOAD(8196,63,248)
290 END
300 ! Datas
310 DATA 0,0,0,0,0,0,0,0,0,0
,0,0,203,20,203,53
320 DATA 203,78,203,231,204,
71,204,150,204,228,205,29,20
5,75,205,96
330 DATA 33,131,35,253,38,18
4,40,183,41,182,42,195,43,19
3,44,179
340 DATA 45,194,47,196,58,18
1,59,180,60,191,61,190,62,19
2,94,197
350 DATA 255,58,58,130,65,84
,240,71,79,133,73,70,132,79,
78,155
360 DATA 79,82,186,80,5,0,16
,0,0,0,33,0,0,0,6,0
370 DATA 16,0,0,0,0,0,96,3,6
7,83,49,0,8,15,36,0
380 DATA 2,224,36,248,216,32
,37,97,37,56,4,192,4,195,2,1
390 DATA 0,1,2,2,37,56,4,32,
32,20,2,0,15,0,2,1
400 DATA 37,72,2,2,0,10,4,32
,32,36,2,0,15,9,2,1
410 DATA 37,56,2,2,0,16,4,32
,32,36,2,10,15,9,200,10
420 DATA 131,86,4,32,38,152,
0,8,22,8,2,1,15,0,200,1
430 DATA 131,28,208,32,37,98
,4,32,32,52,2,0,15,0,2,1
440 DATA 37,82,2,2,0,13,4,32
,32,36,2,0,16,2,2,1
450 DATA 37,4,2,2,0,2,4,32,3
2,44,2,0,15,6,2,38
460 DATA 0,6,4,32,32,36,216,
32,37,92,131,74,216,32,37,93
470 DATA 131,75,216,32,37,94
,131,76,2,0,15,13,200,0,131,
86
480 DATA 2,0,0,3,200,0,131,8
4,4,224,131,208,216,32,37,96
490 DATA 131,109,216,32,37,9
5,131,124,2,0,36,244,2,1,4,9
6
500 DATA 204,1,2,1,38,130,19
6,1,4,32,38,64,0,61,216,32
510 DATA 37,95,131,124,2,224
,131,224,4,96,0,112,37,24,38
,70
520 DATA 36,50,200,62,131,23
6,216,32,152,2,38,68,216,32,
152,2
```

```
530 DATA 38,69,6,32,38,68,2,224,131,224,2,13,152,0,209,32
540 DATA 131,115,9,132,2,36,131,2,2,5,36,242,205,5,2,5
550 DATA 0,58,197,5,6,196,216,4,131,115,4,96,0,96,6,96
560 DATA 131,115,2,224,37,24,216,32,38,68,156,2,216,32,38,69
570 DATA 156,2,3,128,37,24,38,168,32,0,6,160,36,52,193,32
580 DATA 34,158,46,170,193,126,83,224,38,156,192,32,131,86,194,64
590 DATA 2,41,255,248,4,32,32,40,208,193,9,131,7,4,2,2
600 DATA 38,158,5,128,5,132,128,196,19,6,4,32,32,40,220,129
610 DATA 152,1,38,166,22,246,193,4,19,71,2,132,0,7,21,68
620 DATA 4,224,131,208,200,4,131,84,5,132,168,4,131,86,2,224
630 DATA 131,224,4,193,2,12,15,0,195,12,19,1,30,0,2,44
640 DATA 1,0,4,224,131,208,2,140,32,0,19,44,200,12,131,208
650 DATA 29,0,2,2,64,0,152,18,38,167,22,238,2,34,0,8
660 DATA 16,3,192,160,131,210,29,0,192,146,19,230,200,2,131,210
670 DATA 5,194,194,114,209,96,131,85,19,9,156,133,22,242,9,133
680 DATA 2,6,38,158,156,182,22,237,6,5,22,252,5,129,6,153
690 DATA 16,232,30,0,2,224,37,24,192,9,4,32,32,40,9,209
700 DATA 22,4,3,128,2,224,37,24,4,193,6,193,215,65,243,224
710 DATA 38,156,3,128,0,0,0,0
```



```
*        **********
*       ************
*       **        **
*       ** DSKCS1 **
*       **        **
*       ************
*        **********
*
*       (C) M. Eichhorn, Goethestr. 7, 6242 Kronberg/Ts.
*
*       Programm zum Ueberspielen von PROGRAM-Files
*       von Diskette auf Cassette.
*
*       Es werden Standard-Programfiles mit einer max.
*       Laenge von 8K verarbeitet; das Format dieser
*       Files entspricht denen, die mit der SAVE-Utility
*       des EDITOR/ASSEMBLER erzeugt werden.
*
*       Benoetigte Konfiguration: XBASIC
*                                 32K-RAM
*                                 Diskettenlaufwerk
*                                 Cassettenrekorder
*
*       Aufruf des Programms mit:
*
*            CALL LINK("DSKCS1",Dateiname)
*
*
        AORG >24F8
*
        DEF  DSKCS1
*
STRREF  EQU  >2014
VSBW    EQU  >2020
VMBW    EQU  >2024
VSBR    EQU  >2028
VMBR    EQU  >202C
ERR     EQU  >2034
*
NAMPTR  EQU  >8356
*
MYWS    BSS  32
SUBWS   BSS  32
*
NAMBUF  BSS  16
*
PABBUF  DATA >0500          Opcode fuer LOAD
        DATA >1000          VDP-Bufferadresse
        DATA >0000
        DATA >2100          max. >2100 Bytes laden
        DATA >0000
*
PABBF1  DATA >0600          Opcode fuer SAVE
        DATA >1000          VDP-Bufferadresse
        DATA >0000
        DATA >0000
        BYTE >60            Screenoffset fuer Basic
        BYTE >03            Namenlaenge
        TEXT 'CS1'          Devicename
*

B00     BYTE >00
B08     BYTE >08
B0F     BYTE >0F
B24     BYTE >24
*
        EVEN
*
DSKCS1  LWPI MYWS           Eigenen Workspace laden
        MOVB @B0F,@NAMBUF Namenlaenge fuer @STRREF setzen
        CLR  R0
        CLR  R3
        LI   R1,1
        LI   R2,NAMBUF
        BLWP @STRREF        Dateinamen von Basic in NAMBUF uebernehmen
*
PABSET  LI   R0,>0F00       PAB erstellen
        LI   R1,PABBUF
        LI   R2,>000A
        BLWP @VMBW
*
        LI   R0,>0F09       Namenlaenge u. Namen aus NAMBUF in PAB
        LI   R1,NAMBUF
        LI   R2,>0010
        BLWP @VMBW
*
        LI   10,>0F09       Namenlaengenzeiger setzen
        MOV  10,@NAMPTR
        BLWP @DSRLNK        File laden
        DATA 8
        JNE  NOERR          Kein Fehler --> weiter
*
        LI   R1,>0F00       PAB-Startadresse in @>831C
        MOV  R1,@>831C
        MOVB @B24,R0        I/O-Error
        BLWP @ERR           Ruecksprung mit Fehlermeldung
*
NOERR   LI   R0,>0F00       PAB1 erstellen
        LI   R1,PABBF1
        LI   R2,13
        BLWP @VMBW
*
        LI   R0,>1002       Laenge des geladenen Files holen
        LI   R1,MYWS+12
        LI   R2,2
        BLWP @VMBR
*
        LI   R0,>0F06
        AI   R6,6           Laenge des Files + Headerlaenge (6 Bytes)
        BLWP @VMBW          Filelaenge in PAB1 schreiben
*
        MOVB @PABBF1+10,@>834A Devicenamen in FAC schreiben
        MOVB @PABBF1+11,@>834B
        MOVB @PABBF1+12,@>834C
*
        LI   R0,>0F00+13  Pointer auf Devicenamenlaenge
        MOV  R0,@NAMPTR
        LI   R0,>0003       Namenlaenge
```

```
        MOV  R0,@>8354
        CLR  @>83D0        Search-Pointer loeschen
        MOVB @B08,@>836D   DSR-Byte
        MOVB @B00,@>837C   GPL-Statusbyte loeschen
*
        LI   R0,>24F4      GPL-Ruecksprungdatas setzen (fuer GPLLNK)
        LI   R1,>0460      (Mnemonic fuer BRANCH)
        MOV  R1,*R0+
        LI   R1,GPLRT      (Ruecksprungadresse)
        MOV  R1,*R0
*
        BLWP @GPLLNK       Cassettenroutine aufrufen
        DATA >003D
*
        MOVB @B00,@>837C   GPL-Statusbyte loeschen
        LWPI >83E0         GPL-Workspace laden
        B    @>0070        Ruecksprung ins Basic
*
* UNTERPROGRAMME
* =============
*
GPLLNK DATA SUBWS,GPL1
GRMADR BSS  2
*
GPL1    MOV  *R14+,@>83EC
        MOVB @>9802,@GRMADR
        MOVB @>9802,@GRMADR+1
        DEC  @GRMADR
        LWPI >83E0
        LI   R13,>9800
        MOVB @>8373,R4
        SRL  R4,8
        AI   R4,>8302
        LI   R5,>24F2
        MOV  R5,*R4+
        LI   R5,>003A
        MOV  R5,*R4
        SWPB R4
        MOVB R4,@>8373
        B    @>0060
GPLRT   DECT @>8373
        LWPI SUBWS
        MOVB @GRMADR,@>9C02
        MOVB @GRMADR+1,@>9C02
        RTWP
*
DSRLNK DATA SUBWS,DSR1
*
DSRMSK DATA >2000
DSRBUF BSS  8
POINT  TEXT '.'
HAA    BYTE >AA
*
DSR1    MOV  *R14+,R5      Data fuer DSRLNK uebernehmen
        SZCB @DSRMSK,R15   EQU-Bit im Statusregister ruecksetzen
        MOV  @>8356,R0     Zeiger auf Namenlaenge im PAB
        MOV  R0,R9
        AI   R9,>FFF8      Zeiger auf Beginn PAB
        BLWP @VSBR
        MOVB R1,R3
        SRL  R3,8          Namenlaenge in R3
        SETO R4
        LI   R2,DSRBUF
DSR2    INC  R0            Naechstes Byte des Namens
        INC  R4
        C    R4,R3         Ganzer Name verglichen?
        JEQ  DSR3          Ja --> weiter
        BLWP @VSBR
        MOVB R1,*R2+       1 Byte in aus PAB in Buffer
        CB   R1,@POINT     Byte = '.'?
        JNE  DSR2          Nein --> weiter
DSR3    MOV  R4,R4         Devicenamenlaenge = 0?
        JEQ  DSRER1
        CI   R4,7          Devicenamenlaenge groesser 7?
        JGT  DSRER1
        CLR  @>83D0        ROM-Searchpointer loeschen
        MOV  R4,@>8354     Pointer fuer DSR (Devicenamenlaenge)
        INC  R4
        A    R4,@>8356     Pointer fuer DSR (1. Byte nach Devicename)
        LWPI >83E0         GPL-Workspace laden
        CLR  R1            DSR-Nummer loeschen
        LI   R12,>0F00     Zeiger fuer Suche nach DSR
NOROM   MOV  R12,R12
        JEQ  ROMON
        SBZ  0             Deeice ausschalten
ROMON   AI   R12,>0100     naechstes Device
        CLR  @>83D0        ROM-Searchpointer loeschen
        CI   R12,>2000     letztes Device erreicht?
        JEQ  DSRERR
        MOV  R12,@>83D0    ROM-Searchpointer
        SBO  0             Device einschalten
        LI   R2,>4000
        CB   *R2,@HAA      Headerbyte >AA vorhanden?
        JNE  NOROM         Nein --> kein ROM da
        AI   R2,8          Zeiger auf DSRs
        JMP  DSR5
DSR4    MOV  @>83D2,R2     naechste DSR-Startadresse holen
        SBO  0             Device einschalten
DSR5    MOV  *R2,R2        Zeiger auf naechste DSR holen und ...
        JEQ  NOROM         Adresse 0 --> keine DSR
        MOV  R2,@>83D2     ... sichern
        INCT R2
        MOV  *R2+,R9       DSR-Startadresse
        MOVB @>8355,R5     Device/Namenlaenge holen
        JEQ  DSR7
        CB   R5,*R2+       Devicenamenlaenge vergleichen
        JNE  DSR4
        SRL  R5,8          Devicenamenlaenge (16 Bit)
        LI   R6,DSRBUF
DSR6    CB   *R6+,*R2+     Devicenamen vergleichen
        JNE  DSR4
        DEC  R5
        JNE  DSR6
DSR7    INC  R1            Nummer der DSR erhoehen
        BL   *R9           DSR anspringen
        JMP  DSR4          Ruecksprung ohne INCT R11: naechste DSR
        SBZ  0             normaler Ruecksprung: Device ausschalten
        LWPI SUBWS
        MOV  R9,R0
        BLWP @VSBR         PAB-Byte 1 lesen (Statusbyte)
        SRL  R1,13         Fehlercode in Bit 13-15
        JNE  DSRER2
        RTWP
DSRERR  LWPI SUBWS         alten Workspace laden
DSRER1  CLR  R1            Fehler-Code 0 --> DSR nicht gefunden
DSRER2  SWPB R1
        MOVB R1,*R13       Fehlercode in R0
        SOCB @DSRMSK,R15   EQU-Bit im Statusregister (R15) setzen
        RTWP
*
        END
*
```



Club-Ecke

****

Nachdem nun mit der HCA die letzte überregionale deutschsprachige Zeitschrift an den Kiosken verschwindet, in der der TI-99/4A einen festen Platz hatte, wird es für die User dieses Computers nicht leicht werden, zum Beispiel an Informationen über neue Soft- und Hardware zu kommen oder Kontakte zu anderen Usern aufzunehmen.

Doch es gibt auch weiterhin eine Möglichkeit, um ständig auf dem neuesten Stand zu sein und bei allen möglichen Fragen und Problemen mit fachmännischen Antworten und Hilfe rechnen zu können. Die Lösung dieses Problems sind die TI-Clubs, die sich in den Jahren, nachdem die Firma Texas Instruments den Bau von Homecomputern aufgegeben hatte, recht zahlreich gebildet haben. Aus diesen Clubs kamen in den letzten Jahren viele gute, neue Programme und auch Hardware-Erweiterungen, die zu „Texas-Zeiten" nicht denkbar waren.

Viele dieser Clubs haben auch gute Verbindungen in die USA, wo der TI-99/4A und auch der Nachfolger, der Geneve der Firma Myrac, bei weitem nicht so ein Schatten-Dasein führen wie bei uns in Europa. Auch von dort kamen und kommen heute noch Hard- und Software-Spitzenprodukte. Man denke hier nur zum Beispiel an die Programme „DISKASSEMBLER", „EXPLORER" oder „ADVANCED DIAGNOSTIC", die ja im TI-Teil der HCA ausführlich besprochen wurden. Bei der TI-Hardware aus den Staaten hat sich in den letzten Monaten besonders die HORIZON-Ramdisk sehr schnell bei den TI-99/4A-Nutzern durchgesetzt. Und ohne diese User-Clubs wären solche TI-Highpoints vielleicht gar nicht nach Deutschland gekommen.

Jetzt sind diese Clubs hier bei uns das letzte Bindeglied, das die TI-User, die ihrem Computer trotz al-

ler Widrigkeiten treu bleiben wollen (und das sind doch recht viele), zusammenhalten kann.
Wenn auch Sie weiterhin „dabei"-bleiben wollen, so sollten Sie vielleicht überlegen, ob Sie sich nicht einer dieser Interessengemeinschaften anschließen sollten. Sie müssen ja nicht unbedingt als zahlendes Mitglied voll einsteigen, auch ein loser Kontakt zu einer (oder auch mehrerer) dieser Gruppen dürfte mit Sicherheit für beide Seiten von Vorteil sein.
Um Ihnen die Entscheidung hierzu etwas zu erleichtern, finden Sie an dieser Stelle noch einmal eine Reihe von Kontaktadressen verschiedener Clubs, die sich bisher aktiv an der Gestaltung des TI-Teils dieser Zeitschrift beteiligt haben, die sich auch in Zukunft dem TI-99/4A widmen wollen und die vor allen Dingen bereit sind, ihr Know-how zu *fairen* Bedingungen an andere TI-User weiterzugeben.

TI-99/4A User Club Berlin
Franz Neudert
Meinekestraße 25
1000 Berlin 15

Texas Instruments Club Kiel
c/o Michael Rittweger
Nissenstraße 12
2300 Kiel 14

TI-User Group Bremen
c/o Gerhard Hollnagel
Alfred-Faust-Straße 4
2800 Bremen 61

Panorama Computer Club
– TI-99/4A Service –
c/o Hans Greiffenberg
Großglocknerstraße 45
4100 Duisburg 28

SYSTEM-99 User Group
Christopher Winter
Rembrücker Weg 44
6053 Obertshausen

oder
Hartmut Klonz
Homburger Landstr. 143a
6000 Frankfurt 50

oder
Gerd Weißmann
Posenerstraße 2
8000 München 81

TI-Workshop Wiesbaden
Horst Wiese
Eleonorenstraße 6
6200 Wiesbaden

oder
Jürgen Stelter
Berliner Straße 2
6104 Seeheim/Jugendheim

TI-Club ERRORFREE
c/o Manfred Kraus
Saarstraße 127
7254 Hemmingen

oder
c/o Heinz Lücke
Mörikestraße 30
7147 Hochdorf/Enz

TI-99/4A & GENEVE
Workshop Würzburg
Markus Krämer
Leo-Weismantel Straße 1, Apt. 3617
8700 Würzburg

Diese Liste erhebt keinen Anspruch auf Vollständigkeit. Wir werden auch in den nächsten Ausgaben (auf Diskette) die Clubecke weiterführen. Sollten Sie keine Adresse in Ihrer Nähe finden, so setzen Sie sich mit einer der obigen Adressen in Verbindung. Denn da viele Mitglieder dieser Clubs im weiteren Umkreis verstreut sind, gibt es vielleicht auch TI-User in Ihrer Nähe. Also nur Mut. Wir bitten Sie aber, bei Anfragen jeder Art immer einen frankierten Rückumschlag beizulegen. Auch das 3. internationale TI-User-Treffen am 15.10.1988 in Duisburg bietet Gelegenheit, um Kontakte zu anderen Usern zu knüpfen. Es werden sicher alle TI-Clubs aus dem Bundesgebiet hier vertreten sein.

Jürgen Stelter □

TI-TREFFEN 1988

Wie bereits in der letzten Ausgabe angekündigt, richtet in diesem Jahr der Panorama Computer Club in Duisburg das 3. internationale TI-User-Treffen aus. In der Zwischenzeit stehen auch Zeitpunkt und Ort fest.
Auch legt der PCC Wert auf die Feststellung, daß dieses Treffen als reines TI-User-Treffen geplant ist und auch auf jeden Fall so und nicht anders durchgeführt wird.
Das Treffen findet am 15. Oktober 1988 in der
Jugendherberge
Duisburg-Wedau,
Kalkweg 148a
4100 Duisburg 26
(Wedau)
Tel.: 0203/724164
statt.
Hier sind auch Übernachtung und Verpflegung zu erschwinglichen Preisen möglich.
Übernachtung mit Frühstück pro Person und Nacht 15 Mark, bei Voranmeldung und Bezahlung bis 30.6.88 reduziert sich dieser Preis auf 13 Mark. Bitte Kopie der Überweisung beilegen! Kto.-Nr. 234-000056 (BLZ 34050000) bei der Stadtsparkasse Duisburg).
Bettwäsche muß mitgebracht oder kann gegen eine Gebühr von 3,50 Mark ausgeliehen werden.
Diejenigen, die ein Hotel vorziehen, erhalten auf Anfrage eine Hotelliste. Diese liegt auch den Einladungen bei, die die TI-Clubs vom PCC erhalten haben. Hier empfiehlt sich eine frühzeitige Reservierung.
Noch ein paar Worte zum Ablauf dieses Treffens.
Freitag, 14.10.:
Aufbau der Anlagen ab ca. 18.00 Uhr möglich.
Samstag, 15.10.:
Aufbau der Anlagen bis ca. 10.00 Uhr.
10.00 Uhr bis 18.00 Uhr offizieller Teil des Treffens mit Publikum.
Ab ca. 20.00 Uhr gemütliches Beisammensein der TI-Club-Mitglieder.
Sonntag, 16.10.:
8.00 Uhr bis 12.00 Uhr Zeit für Club-Interna, ab ca. 12.00 Uhr Abbau und Ende des Treffens.

Der PCC hat mit 35 TI-Clubs Verbindung aufgenommen und erwartet eine große Beteiligung. Geplant sind u.a. eine elektronische Börse sowie verschiedene Videobeiträge über die Arbeit in den TI-Clubs. Auch ist ein Programmier-Wettbewerb geplant, bei dem jedoch keine Einzelpersonen, sondern nur Gemeinschaftsbeiträge zugelassen sind.
Ausführliche Informationen (Anreisebeschreibung, Hotelverzeichnis, Tischreservierung etc.) sind gegen einen an sich selbst adressierten, mit 1,20 Mark frankierten Umschlag (DIN A4) zu erhalten beim
PANORAMA COMPUTER CLUB
– TI-99/4A Service –
c/o Hans Greiffenberg
Großglocknerstraße 45
D-4100 Duisburg 28
Telefon: 0203/724475 □

„RAM-ANZEIGE" ENTHÄLT UNNÖTIGE DATAS

In der HCA 12/1 '88 wurde auf den Seiten 55/56 die Pokeliste zu dem Programm „RAM-Anzeige" veröffentlicht.
Bei einem etwas genaueren Studium des Listings fiel mir auf, daß in den DATA-Zeilen einiges überflüssig ist. Wenn Zeile 310 geändert wird zu ADR=9492, dann können die DATA-Zeilen 350 und 360 sowie 450 bis 490 entfallen. Aus der Zeile 440 kann alles außer den ersten vier Werten entfernt werden. Das Programm arbeitet dann trotzdem noch einwandfrei.
H. Fiedler,
Baar

Das Programm „RAM-Anzeige" wurde ganz offensichtlich mit einem automatisch arbeitenden Pokelistengenerator erstellt, der auf einen zu großen Bereich eingestellt wurde. Von daher ist es durchaus möglich, daß da einiges überflüssig ist.

All Systems go! (Ende)

Nachdem in den ersten drei Folgen der Serie Diskettensysteme im allgemeinen und das des TI 99/4A recht ausführlich in der Theorie behandelt wurden, sollen diesmal auch die User „mit dem warmen Lötkolben" zu ihrem Recht kommen.

Darum finden Sie in dieser Ausgabe noch eine Platinenvorlage vor. Es wurde aus leicht zu begreifenden Gründen eine doppelseitige Version im Format 150 x 200 gewählt. Auf den ersten Blick scheint dies etwas großzügig zu sein und auf den zweiten wird dieser Eindruck verstärkt. Verstehen Sie dies bitte als Zugeständnis an Sie, denn dadurch sind Sie in der Lage, auf Ober- und Unterseite der Platine Teile anlöten zu können, ohne daß man diese durchkontaktieren muß. Verwenden Sie für die ICs Präzisionsfassungen, die man auch gut von oben erreicht. Der Aufwand der Schaltung ist minimal und bringt Ihnen folgende Funktionen: Der SHUGART-BUS in einer DAISY-CHAIN-Realisation (vergleiche dazu die Abhandlung in diesem Artikel), wie man sie beim TI findet, wird mit banaler Logik und geringem Aufwand in eine RADIAL-CHAIN-CONNECTION umgewandelt. Für *jedes* Drive steht Ihnen somit ein kompletter Bus zur Verfügung, der für alle Signale einen eigenen Treiber aufweist.
Was das Ganze soll? Nun, zum einen können Sie die Verbindung des Kontrollerausgangs am TI (beziehungsweise der BOX) und dieser Platine 3 bis 5 Meter lang machen und damit, ohne das Risiko von Datenverlusten oder Kontrollerdefekten, auch mechanisch nicht mehr in der Box unterzubringende Laufwerke (oder auch vier alte Drives mit voller Höhe) bequem an den Rechner anschließen, da Sie ja jetzt ein Gehäuse Ihrer Wahl in die Landschaft stellen können. Zum anderen ist die realisierte Anordnung für Experimente wie geschaffen, da kein teurer Defekt auftreten kann. Und zum dritten haben Sie nun die Möglichkeit, bis zu 16 Laufwerke an den Kontroller anzuschließen, meinetwegen auch noch mehr. Jeder Buxausgang auf der Platine kann nämlich seinerseits wieder als DAISY-CHAIN-Ausgang betrachtet werden.
Die Platine weitet den Bus in vier Busse auf, wobei jeder Bus alle SHG-Bus-Signale ausgibt und alle annimmt (auch READY, damit Sie die Schaltung auch an TI-fremden Kontrollern mit SHG-BUS verwenden können), mit der Einschränkung, daß jeder 34-polige Ausgangsport nur eines der vier Drive-Select-Signal-Leitungen besitzt. Ein Port für Drive 0, usw. Sicherheitshalber wurde auf vier Ports ausgelegt. Haben Sie vier Laufwerke, die Sie anschließen wollen, führen Sie den Bus jedes Laufwerks an den Port der Platine. Die Programmierung der Laufwerke ist dann nur noch unter der gleichlautenden Portnummer möglich. Das Drive an Port 0 muß demnach als Drive 0 gesteckt werden. Und jedes Laufwerk behält seinen Widerstandsarray. Bei alten Drives, die sich nicht als Drive 3 (4. Laufwerk) ansprechen lassen, löten Sie die Select-Leitung an Port 4 einfach auf dem Buspin, der sonst Drive 2 (3. Drive) anspricht. Da ja Drive 2 seinen eigenen Ausgang hat, kann es nicht zum Ansprechen von zwei Laufwerken unter der gleichen Nummer kommen.

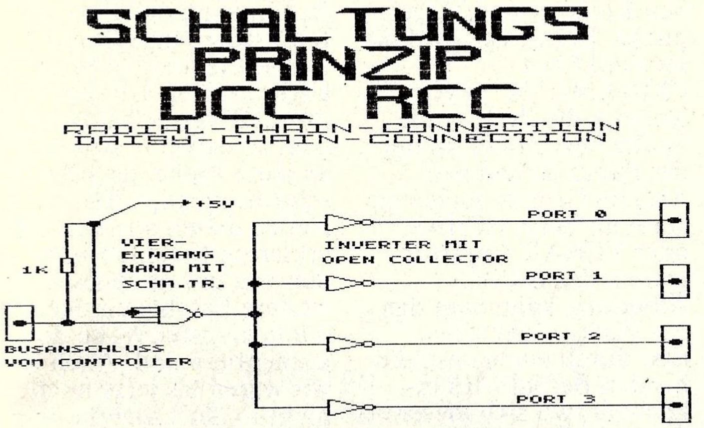
BESTÜCKUNGSPLAN DCC → RCC

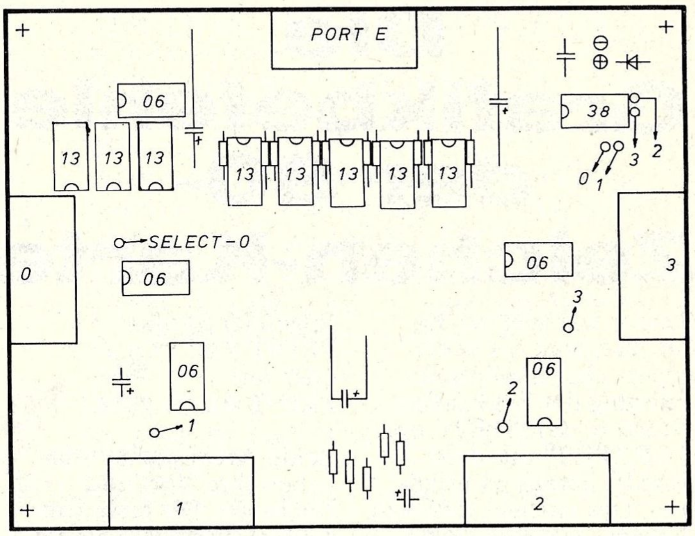
Bauteil	Wert	Bemerkungen
R1 bis R15	1 kOhm/ 0.25 Watt	Kohleschichtwiderstände in handelsüblicher Ausführung.
C1	1 nF/ 63 Volt	Keramischer Scheibenkondensator.
C2 bis C3	470 µF/ 16 Volt (axial)	Elektrolythkondensator.
C4	1000 µF/ 16 Volt (radial)	Elektrolytkondensator.
C6 bis C6	100 µF/ 16 Volt (radial)	Elektrolytkondensator.
IC1 bis IC8	74...13	2 x Nandgatter mit 4 Eingängen.
IC9	74...38	4 x Nandgatter mit 2 Eingängen.
IC10 bis IC14	74...06	6fach Open Collector Inverter.
D1	1 N 4007	1A-Siliziumdiode.
Port E und Port 0 bis 3	34pol. Pfostenstecker, 90 Grad für Platinenmontage, hier: 5 Stück.	

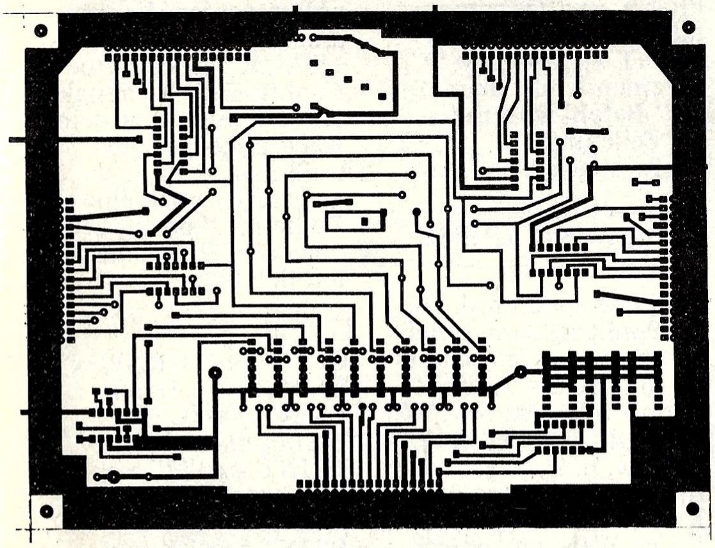
Sonstiges: 34poliges Flachbandkabel (AWG 28), Länge nach Bedarf (siehe Text).
5 Stück Pfostenbuchsen 34polig für obiges Flachbandkabel in Schneid-/Klemmtechnik.
Stecker für Laufwerke nach Bedarf.
1 x Platine.

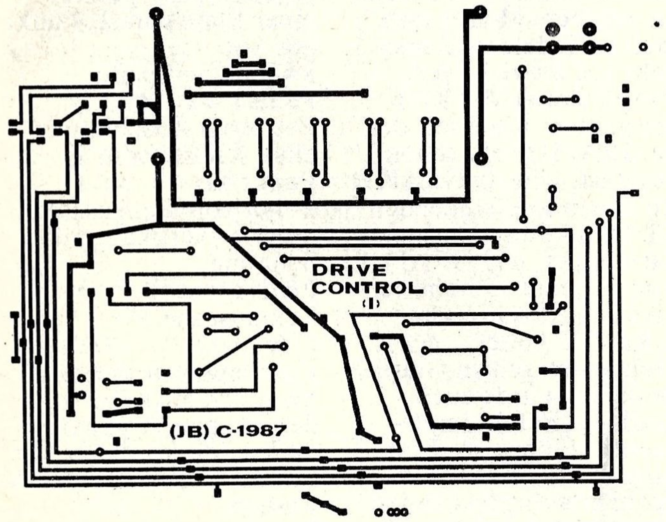
LAUFWERKS-GRUPPE

Der Gag ist die Möglichkeit, an jeden Port eine ganze Laufwerksgruppe anzuschließen. Dabei erhält nur ein Laufwerk dieser Gruppe den Terminator, eben wie bei der normalen DAISY-CHAIN.
Das Laufwerk, bei dem der Jumper für die Programmierung gesteckt ist, wird angesprochen. Legen Sie diese 1-polige Brückenfunktion auf ei-







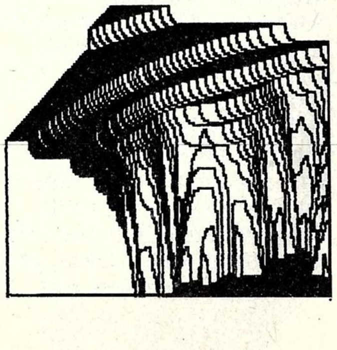


nen frontseitig montierten Schalter, können Sie per Knopfdruck zwischen verschiedenen Laufwerken umschalten, die vom TI alle unter der gleichen Nummer angesprochen werden. Das Pünktchen auf dem I ist dann noch eine programmgesteuerte Version, die den Schalter ersetzt und eben über ein Programm und gegebenenfalls Relais zum Entkoppeln angesprochen wird. Hier liegt eine sehr einfache, aber gewinnbringende Einsatzform der Acht-Kanal-Ein-/Ausgabeeinheit (siehe TI-REVUE). Ich persönlich habe damit eine Einheit erstellt, bei der vier 3-Zoll-, vier 3.5-Zoll- und vier 5.25-Zoll-Drives gleichzeitig am Rechner hängen und ich im Bedarfsfall sowohl manuell als auch per Programm auf ein anderes Format umschalten kann. Wenn Sie diese Umschaltlogik interessiert, schreiben Sie dem Verlag. Ich melde mich auf Anfragen immer.

KONTROLLE ENTLASTEN

Der Kontroller des TI kann nun etwas entlastet werden. Auf der Platine wird das Kabel nur mit 1 kOhm terminiert, gegenüber 150 Ohm bei der Standardanwendung. Somit reduziert sich der Busstrom von 40 Milliampere auf 0.005 Ampere, etwas über 10 Prozent des alten Wertes. Und die geringeren Stromspitzen lassen den Kontroller eben länger leben. Die Eingangsterminierung der Platine (das, was sonst im Kontroller sitzt), kann von Ihnen je nach Erfordernis festgelegt werden. Fangen Sie bei 1.2 kOhm an und reduzieren Sie bei Nichtfunktion herunter bis maximal 150 Ohm. Dann *muß* es aber klappen.
Die Platine wird mit +5 V (±5%) gegen Masse, genau wie die 5-Volt-Versorgung der Laufwerke, versorgt, wobei die Stromaufnahme von der verwendeten Technologie abhängt. Sie können alles aus der 74er-Reihe, sofern es funktions- und anschlußkompatibel ist, verwenden, seien es nun TTL-Standard- oder modernste 74HCT-Typen. Wie gesagt, ist die Schaltungstechnik banal. Daher wurde nur ein reduzierter Schaltplan erstellt, der den Verlauf einer Leitung zeigt. Nach Studium dieser Artikelserie liegt alles andere auf der Hand. Bisher wurde mit dem Wissen um die Diskettenantriebe den Usern viel Geld aus der Tasche gezogen. Nun sollten sich auch die Sparer voll ausstatten können. Zum Schluß noch einmal: Wenn Sie Fragen haben, stellen Sie sie!
Niemand kann Ihnen antworten, wenn man die Probleme der User nicht kennt. Und was glauben Sie wohl, wozu der Verlag so viel Papier bedrukken läßt?
. . . TO SATISFY THE NEED!

J. Brilmayer □



Die Grafikbefehle der 80-Zeichen-Karte

Leider war von der Fa. Mechatronik bis heute nichts über die Programmierung der beiden Grafikmodi „GRAPH1" und „GRAPH2" aus dem BASIC heraus zu erfahren. Das einzige, was man bekam, war eine Demo-Diskette mit einem Programm, das man laut Mechatronik analysieren solle, dann hätte man schon eine Bedienungsanleitung. Nun, wir haben uns diese Arbeit gemacht und einige der Geheimnisse gelüftet. Hier eine kurze Erläuterung der bisher bekannten Befehle.

Zuerst etwas zur Syntax der einzelnen Befehle:

Die Grafikmodi werden wie auch der TEXT80-Modus als Datei angesprochen, in die dann die einzelnen Befehle geschrieben werden. Diese Befehle bestehen aber nicht wie im Textmodus aus ASCII-Steuerzeichen, sondern aus einzelnen Buchstaben, denen in der Regel noch die Werte der Parameter folgen. Es gibt zwei Grafikmodi im Betriebssystem 2.0 der 80-Zeichen-Karte.
GRAPH1 = dieser Modus bietet eine Auflösung von 512 x 215 Pixel bei 16 Farben.
GRAPH2 = hier ist die Auflösung geringer. 256 x 212 Pixel bei allerdings 256 Farben.
Angesprochen werden diese Modi mit dem Befehl
OPEN #x:("GRAPHy")
Wobei „x" eine beliebige Zahl zwischen 1 und 255 für die Dateinummer, ein „y" eine 1 oder eine 2 für den Modus darstellt.
Alle weiteren Befehle werden nun in diese Datei geschrieben und zwar mit folgender Syntax:
PRINT #x:"x",y,z
oder mit
PRINT #x:"x y z a b c . . . . ".
Beide Arten der Syntax haben ihre Vor- und Nachteile. Die erste Eingabeart erlaubt in einem viel größeren Maße die Verarbeitung von Variablen, ist aber in der Ausführung ziemlich langsam. Der Vorteil der zweiten Befehlsweise ist im wesentlichen deren Geschwindigkeit. Es ist hierbei möglich, einen PRINT-Befehl über die ganze Eingabelänge einer Zeile auszudehnen. Besonders übersichtlich ist das ganze dann allerdings nicht mehr. Vor allem gibt es bei dieser Art der Eingabe einige Probleme mit der Variablenverarbeitung. Für einfache Grafiken ist also auf jeden Fall die zweite Variante zu empfehlen, während die erstere bei komplexen Fällen, wie zum Beispiel dem Ausfüllen einer ganzen Fläche, zum Einsatz kommen sollte.

Zu den einzelnen Befehlen:

Nach der Dateieröffnung muß zuerst festgelegt werden, in welcher Farbe der erste Pixel gesetzt werden soll. Dies geschieht mit dem Befehl
PRINT #1:"C",x
Bei GRAPH1 steht x für einen Wert zwischen 0 und 15, bei GRAPH2 für einen Wert zwischen 0 und 255, entsprechend der Farbmöglichkeiten der beiden Modi.
Bevor man seine Grafik erstellt, muß der erste Pixel gesetzt werden.
Für eine Linie ist das deren Anfangspunkt, für einen Kreis dessen Mittelpunkt. Festgelegt wird dieser Punkt mit
PRINT #x:"M",x,y,
wobei x die Zeile, y die Spalte bezeichnet, in der der Punkt stehen soll.
Also im Grunde nichts anderes als beim HCHAR oder VCHAR Unterprogramm des BASIC.
Allerdings kann man diesen Punkt nicht sehen. Erst durch einen anschließenden Befehl wird erkennbar, wo sich dieser Punkt befindet. Weiter geht es dann mit
PRINT #x:"D",x,y
für eine Linie oder mit
PRINT #x:"W",x,y
für einen Kreis. Beim „D"-Befehl wird mit x die Zeile und mit „y" die Spalte angegeben, in der

LINIEN UND KREISE

die Linie enden soll. Dieser Punkt ist dann gleichzeitig der Ausgangspunkt für die nächste Aktion.
Beim Kreis, also bei „W", wird mit „x" die horizontale, mit „y" die vertikale Ausdehnung des Kreises festgelegt. Hierbei ist allerdings einiges zu beachten. Je nachdem, in welchem Modus man sich befindet, muß man sehr auf die richtigen Proportionen des Kreises achten. Ein Kreis der in GRAPH2 sehr schön rund ist, wird in GRAPH1 zu einem kopfstehenden Ei. Dies kommt ganz einfach daher, daß GRAPH1 eine doppelt so hohe vertikale Auflösung wie GRAPH2 bietet.
Neben den reinen Grafikbefehlen wurde auch noch ein Befehl implementiert, der eine Textausgabe in beiden Grafikmodi erlaubt. Es handelt sich um den Befehl „P".
Die Textausgabe funktioniert dabei ganz einfach:
PRINT #x:"P. . . ."
Die Punkte stehen hier für den auszugebenden Text, der eine beliebige Länge haben kann. In Verbindung mit einer Schleife und dem „C"-Befehl kann der Text laufend seine Farbe ändern.
Selbstverständlich muß auch hier der Startpunkt mit „M" festgelegt werden, so daß eine Ausgabe an jeder Stelle des Bildschirms möglich ist.
Neben diesen ausführlich erklärten Befehlen wurden von uns noch zwei weitere Befehle entdeckt, die in keinster Weise dokumentiert sind. Auch wir waren bis jetzt noch nicht in der Lage, die Syntax und die Parameter zu entschlüsseln, die hier benötigt werden. Es sind dies die Befehle „J" und „R". „R" scheint eine Funktion für Rectangles, also Rechtecke, zu sein.
Aber nichts genaues weiß man eben noch nicht. Hier noch eine Zusammenfassung der oben ausführlich erklärten Befehle:

OPEN #x:("GRAPHx")
Eröffnen der Datei
PRINT #x:"C",x
Festlegen der Farbe
PRINT #x:"M".x.y
Festlegen 1. Pixel
x=Zeile, y=Spalte
PRINT #x:"D",x,y
Festlegen Endpunkt einer Linie vom 1. Punkt aus
x=Zeile, y=Spalte
PRINT #x:"W",x,y
Festlegen Ausdehnung eines Kreises vom 1. Punkt aus
x=horizontale Ausdehnung, y=vertikale Ausdehnung
PRINT #x:"P. . . ."
Textausgabe

Zum besseren Verständnis sollen auch die beiden kleinen Programme dienen, die zu dieser kurzen Anleitung abgedruckt werden.
Wer weitere Anregungen oder auch Fragen zur 80-Zeichen-Karte hat, der melde sich bei der Redaktion, die diese Anfragen sicher weiterleiten wird.

Mike Heuser
TI 99er Workshop Rheinland □

```
10 !  ********************
12 !  *                  *
14 !  *    HIRESDEMO     *
16 !  *                  *
18 !  *   Copyright by   *
20 !  *                  *
22 !  *   MIKE HEUSER    *
24 !  *                  *
26 !  *                  *
28 !  *Benoetigte Geraete*
30 !  *                  *
32 !  * TI99/4A  Konsole *
34 !  *  Extended Basic  *
36 !  * 80-Zeichen Karte *
38 !  *                  *
40 !  *                  *
42 !  * Speicherbelegung *
44 !  *    1977 Bytes    *
46 !  *                  *
48 !  ********************
50 !
80 ! HIGH-RESOLUTION-GRAFIK
     AUF DER 80-ZEICHEN-KAR-
     TE.
81 ! DEMO VON MIKE HEUSER
     04.1988
82 ! ======================
100 OPEN #1:"GRAPH2" !EROEFF
NEN GRAFIKMODUS 256X212 PIXE
L
110 PRINT #1:"M",20,20 !AUSG
ANGSPUNKT ZEILE,SPALTE
120 PRINT #1:"C",30 !FARBE
130 PRINT #1:"D 20 100 D 100
 100 D 100 20 D 20 20" ! DRA
W ZEILE SPALTE ZIELPUNKT FES
TLEGEN
140 PRINT #1:"M",100,100
150 PRINT #1:"D 100 180 D 18
0 180 D 180 100 D 100 100"
160 PRINT #1:"M",100,20
170 PRINT #1:"D 180 20 D 180
 100"
180 PRINT #1:"M",20,100
190 PRINT #1:"D 20 180 D 100
 180"
200 PRINT #1:"M 20 20 C 15"
210 PRINT #1:"D 180 180"
220 PRINT #1:"M",20,180
230 PRINT #1:"D 180 20"
240 PRINT #1:"M 100 20 C 45"
250 PRINT #1:"D 20 100 D 100
 180 D 180 100 D 100 20"
260 PRINT #1:"M 100 100 C 60
" !AUSGANGSPUNKT FUER KREIS
   FESTLEGEN Z,S C=FARBE
270 PRINT #1:"W",56,56 !KREI
S AUSDEHNUNG HOR,VERT.
280 PRINT #1:"M 18 190 PTast
e druecken"
290 CALL KEY(0,K,S):: IF S=0
 THEN 290
300 CLOSE #1
310 CALL CLEAR :: OPEN #1:"G
RAPH1" :: PRINT #1:"C",10
320 PRINT #1:"M",150,80
330 PRINT #1:"D 400 80 D 400
 150 D 150 150 D 150 80 D 27
5 30 D 400 80" !HAUS
340 PRINT #1:"M",165,115
350 PRINT #1:"D 202 115 D 20
2 150 D 165 150 D 165 115" !
 TUERE
360 PRINT #1:"M",166,135
370 PRINT #1:"D",175,135 !
 KLINKE
380 PRINT #1:"M 1 165 D 510
165 M 1 150 D 510 150" !
 STRASSE
390 PRINT #1:"M 250 100 D 35
0 100 D 350 130 D 250 130 D
250 100 M 300 100 D 300 130
M 250 115 D 350 115" !
FENSTER
400 PRINT #1:"M 270 60 W 10
20" !DACHLUKE
410 PRINT #1:"M 60 100 W 20
40 M 48 120 D 48 150 M 72 12
0 D 72 150" !BAUM
420 PRINT #1:"M 6 180 PTaste
 druecken"
430 CALL KEY(0,K,S):: IF S=0
 THEN 430
440 CLOSE #1 :: CALL CLEAR
450 RANDOMIZE
460 OPEN #1:"GRAPH1"
470 PRINT #1:"C",10;"M1 10 P
Sternkarte von M-87":"M1 20
P=================="
480 FOR WH=1 TO 300
490 B=INT(RND*212)+1 :: A=IN
T(RND*512)+1 :: C=INT(RND*16
)+1
500 PRINT #1:"C",C :: PRINT
#1:"M",A,B :: PRINT #1:"D",A
,B
510 NEXT WH
520 PRINT #1:"M 6 180 PTaste
 druecken"
530 CALL KEY(0,K,S):: IF S=0
 THEN 530
540 CLOSE #1 :: CALL CLEAR :
: END
```

```
10 REM*********************
12 REM*                   *
14 REM*     FILLDEMO      *
16 REM*                   *
18 REM*   Copyright by    *
20 REM*                   *
22 REM*   MIKE HEUSER     *
24 REM*                   *
26 REM*                   *
28 REM*Benoetigte Geraete*
30 REM*                   *
32 REM* TI99/4A  Konsole *
34 REM*  Extended Basic  *
36 REM* 80-Zeichen Karte *
38 REM*                   *
40 REM*                   *
42 REM* Speicherbelegung *
44 REM*    1722 Bytes     *
46 REM*                   *
48 REM*********************
50 REM
100 CALL CLEAR
110 OPEN #1:"GRAPH1" :: PRIN
T #1:"C",10 !EROEFFNEN, FARB
E SETZEN
120 PRINT #1:"M 150 60 D 350
 60 D 350 150 D 150 150 D 15
0 60" ! AUSSENRAND
130 PRINT #1:"M 180 75 D 320
 75 D 320 135 D 180 135 D 18
0 75" ! INNENRAND
140 PRINT #1:"C",4 !FARBE F.
FILL
150 FOR S=61 TO 74
160 PRINT #1:"M",151,S :: PR
INT #1:"D",349,S :: NEXT S
170 FOR S=75 TO 135
180 PRINT #1:"M",151,S :: PR
INT #1:"D",179,S :: NEXT S
190 FOR S=75 TO 135
200 PRINT #1:"M",321,S :: PR
INT #1:"D",349,S :: NEXT S
210 FOR S=136 TO 149
220 PRINT #1:"M",151,S :: PR
INT #1:"D",349,S :: NEXT S
230 PRINT #1:"C",12 !FARBE F
.FILL 2
240 FOR S=76 TO 134
250 PRINT #1:"M",181,S :: PR
INT #1:"D",319,S :: NEXT S
260 CALL KEY(0,K,S):: IF S=0
 THEN 260
270 CLOSE #1
280 CALL CLEAR
290 OPEN #1:"GRAPH1" :: PRIN
T #1:"C",10 !EROEFFNEN, FARB
E SETZEN
300 PRINT #1:"M 240 60 D 150
 150 D 330 150 D 240 60" ! A
USSENRAND
310 PRINT #1:"M 240 75 D 173
 142 D 307 142 D 240 75" ! I
NNENRAND1
320 PRINT #1:"M 240 75 D 240
 60 M 150 150 D 173 142 M 33
0 150 D 307 142 " ! INNENRAN
D1
330 PRINT #1:"C",4 !FARBE F.
FILL
340 A,B=240 :: FOR WH=77 TO
141 !ZAEHLSCHLEIFE FUER ZEI-
LEN
350 A=A-1 :: B=B+1 :: PRINT
#1:"M",A,WH :: PRINT #1:"D",
B,WH !FILLROUTINE
360 NEXT WH
370 PRINT #1:"C",12 !FARBE F
.FILL
380 A,B=239 :: FOR WH=63 TO
75
390 A=A-1 :: PRINT #1:"M",A,
WH :: PRINT #1:"D",B,WH :: N
EXT WH
400 PRINT #1:"C",5
410 A=226 :: B=239 :: FOR WH
=76 TO 149
420 A=A-1 :: IF WH>142 THEN
B=B-3 ELSE B=B-1
430 PRINT #1:"M",A,WH :: PRI
NT #1:"D",B,WH :: NEXT WH
440 PRINT #1:"C",6
450 A,B=241 :: FOR WH=63 TO
75
460 B=B+1 :: PRINT #1:"M",A,
WH :: PRINT #1:"D",B,WH :: N
EXT WH
470 PRINT #1:"C",8
480 A=241 :: B=254 :: FOR WH
=76 TO 149
490 B=B+1 :: IF WH>142 THEN
A=A+3 ELSE A=A+1
500 PRINT #1:"M",A,WH :: PRI
NT #1:"D",B,WH :: NEXT WH
510 PRINT #1:"C",9
520 A=174 :: B=306 :: FOR WH
=143 TO 149
530 A=A-3 :: B=B+3
540 PRINT #1:"M",A,WH :: PRI
NT #1:"D",B,WH :: NEXT WH
550 CALL KEY(0,K,S):: IF S=0
 THEN 550
560 CLOSE #1
```

Guten Tag liebe Leser,

Mit dieser Ausgabe der HOME COMPUTER AKTIV halten Sie das letzte Heft dieser Zeitschrift in den Händen, das in der gewohnten Form erscheint Nachdem schon seit einigen Jahren unser TI99/4A nicht mehr hergestellt wird und somit auch die Anzahl der TI-User langsam, aber sicher kleiner wird, haben jetzt auch die Firmen, die dem „Weltstandard MSX zum Durchbruch verhelfen wollten", endgültig das Handtuch geworfen. Somit wird auch bei diesem Computertyp in der nächsten Zeit die Anzahl der User mit Sicherheit deutlich zurückgehen. Und auch bei den kleinen Atari-Computern, die bisher in der HCA vertreten waren, sieht es nicht besser aus. Die ständig fallenden Preise für die immer besseren und leistungsstärkeren neuen Systeme, egal, ob Amiga, Atari oder auch XT oder AT haben dazu beigetragen daß viele unserer ehemaligen Leser ihren Home-Computer gegen einen dieser PCs eingetauscht haben. Dies beweisen auch die ständig steigenden Leserzahlen z.B. der PC-POPULÄR, dem jüngsten Sproß unseres Verlages.

So kann es nicht ausbleiben, daß für eine Zeitschrift wie die HCA der Zeitpunkt kommt, an dem die Herstellungskosten in keinem Verhältnis mehr zu der Auflagenzahl stehen und der Verlag sich schweren Herzens entschließen muß, die notwendigen Konsequenzen zu ziehen.

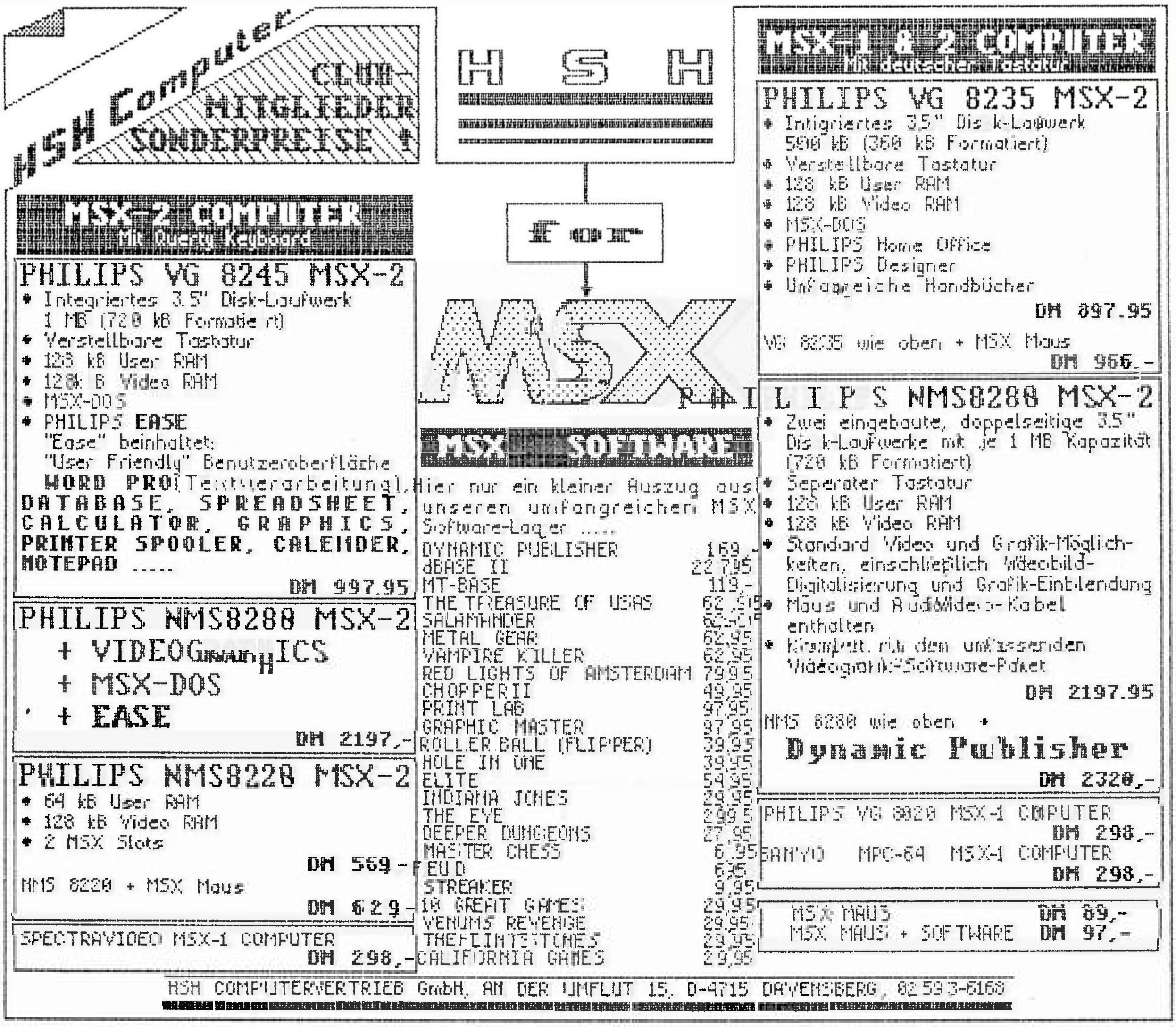
Das heißt: Mit dieser Ausgabe stellt die HCA ihr Erscheinen ein, da die auftretenden Verluste auch die Arbeitsplätze im gesamten Verlag gefährden. Es bleibt uns also nichts anderes, als allen HCA-Lesern noch einmal für die bisherige Treue zu danken. Es hat nicht sollen sein.

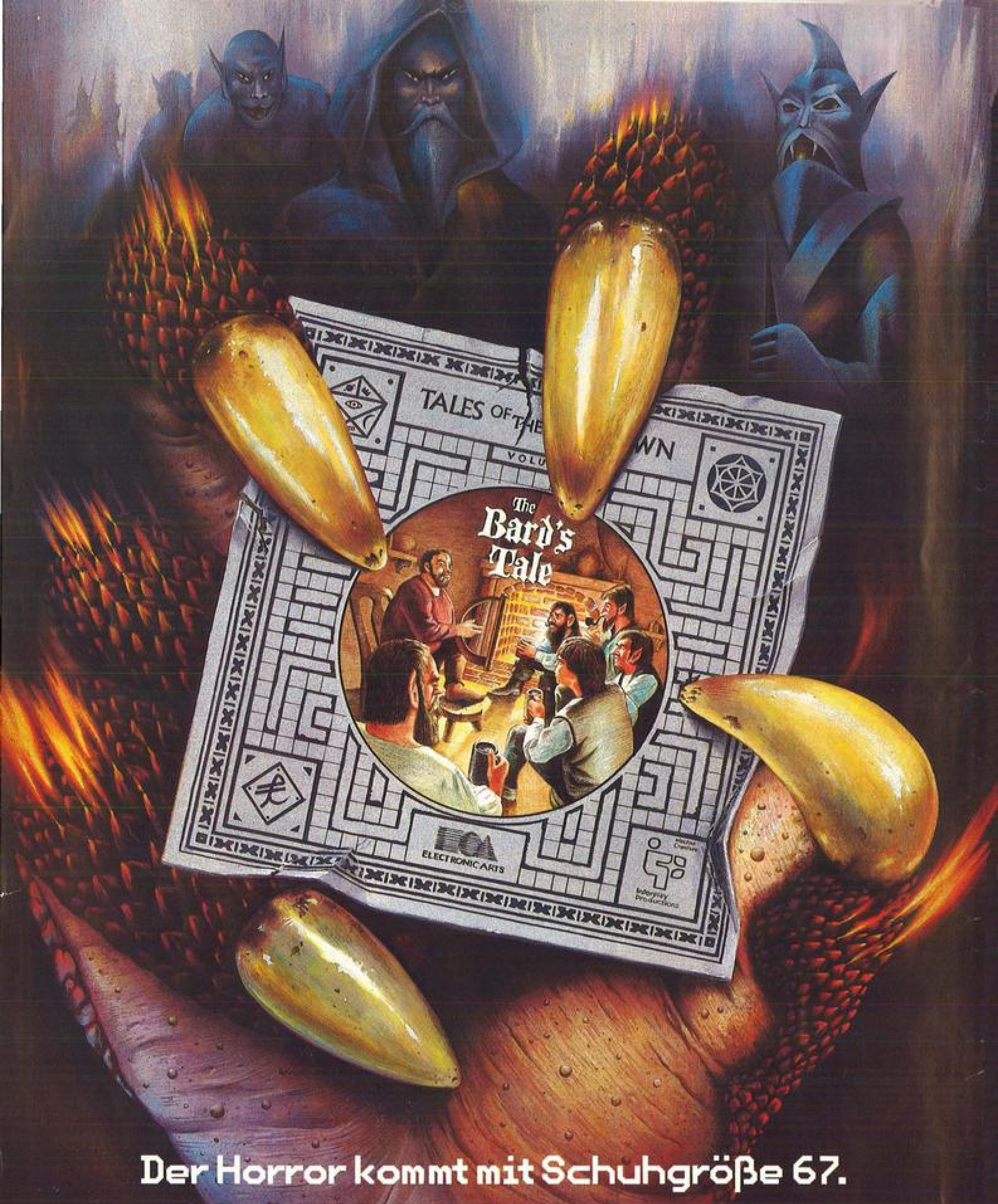
Ihre HCA-Redaktion

*

PS: Die Abonnenten dieser Zeitschrift finden im Inneren einen Brief des Verlages, den wir Ihrer Aufmerksamkeit empfehlen und um dessen Beantwortung der Verlag bittet.

Anzeige





HCA alias Home omputer Aktiv nacque dalla fusione delle riviste TI-Revue e Msx Revue. Con la sua chiusura avvenuta nel 1988 fini' l'epoca delle riviste Europee dedicate al TI99. Ringrazio l'amico Ralf che mi ha fornito questa rarissima rivista con la quale ho completato la raccolta di TI-Revue.

HCA, aka Home Computer Aktiv, was born from the merger of the TI-Revue and Msx Revue magazines. Its closure in 1988 marked the end of European magazines dedicated to the TI99. I thank my friend Ralf for providing me with this very rare magazine, which completed my TI-Revue collection.

Ringrazio altresi' tutti gli amici del gruppo romano 8BitInside e della rivista Retromagazine World che mi hanno supportato in questo lavoro. Se qualcuno vuole contattarmi mi puo' trovare sul forum dediato al TI99 presente sul sito AtariAge.
Astharot

I also thank all my friends from the Roman group 8BitInside and Retromagazine World who supported me in this work. If anyone wants to contact me, they can find me on the TI99 forum on the AtariAge website.
Astharot